Getting Started
- Introduction to Node.js
- How to install Node.js
- How much JavaScript do you need to know to use Node.js?
- Differences between Node.js and the Browser
- The V8 JavaScript Engine
- An introduction to the npm package manager
- ECMAScript 2015 (ES6) and beyond
- Node.js, the difference between development and production
- Node.js with TypeScript
- Node.js with WebAssembly
- Debugging Node.js
- Profiling Node.js Applications
- Security Best Practices

Asynchronous Work
- Asynchronous flow control
- Overview of Blocking vs Non-Blocking
- JavaScript Asynchronous Programming and Callbacks
- Discover JavaScript Timers
- The Node.js Event Loop
- The Node.js Event Emitter
- Understanding process.nextTick()
- Understanding setImmediate()
- Don't Block the Event Loop

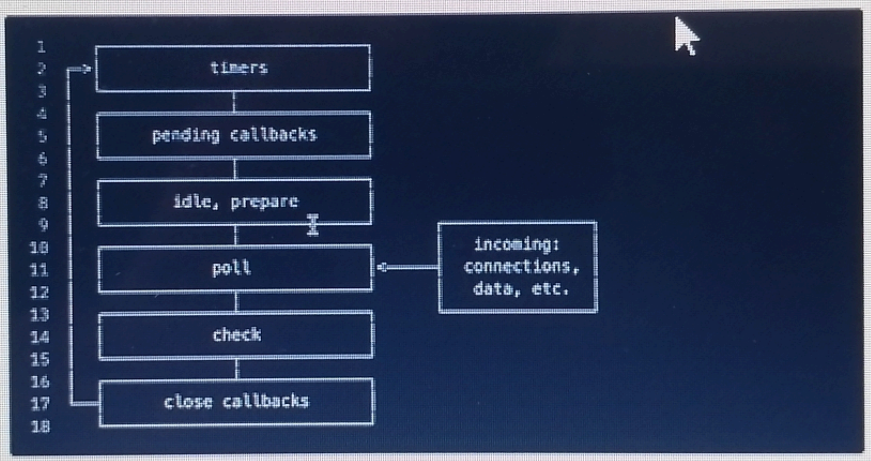
Manipulating Files
- Node.js file stats

in further detail later in this topic.

Event Loop Explained

When Node.js starts, it initializes the event loop, processes the provided input script (or drops into the REPL, which is not covered in this document) which may make async API calls, schedule timers, or call `process.nextTick()`, then begins processing the event loop.

The following diagram shows a simplified overview of the event loop's order of operations.



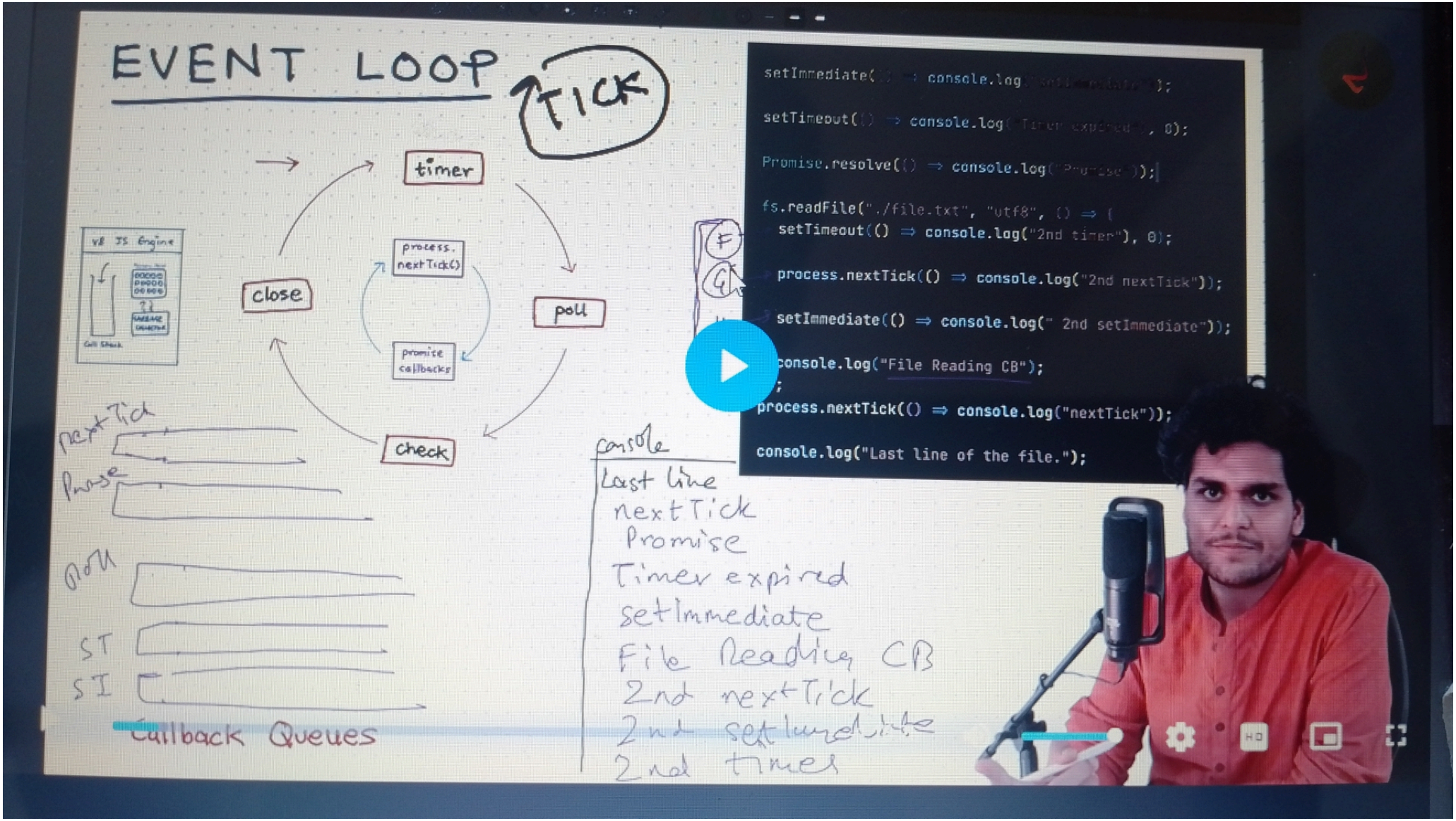
> Each box will be referred to as a "phase" of the event loop.

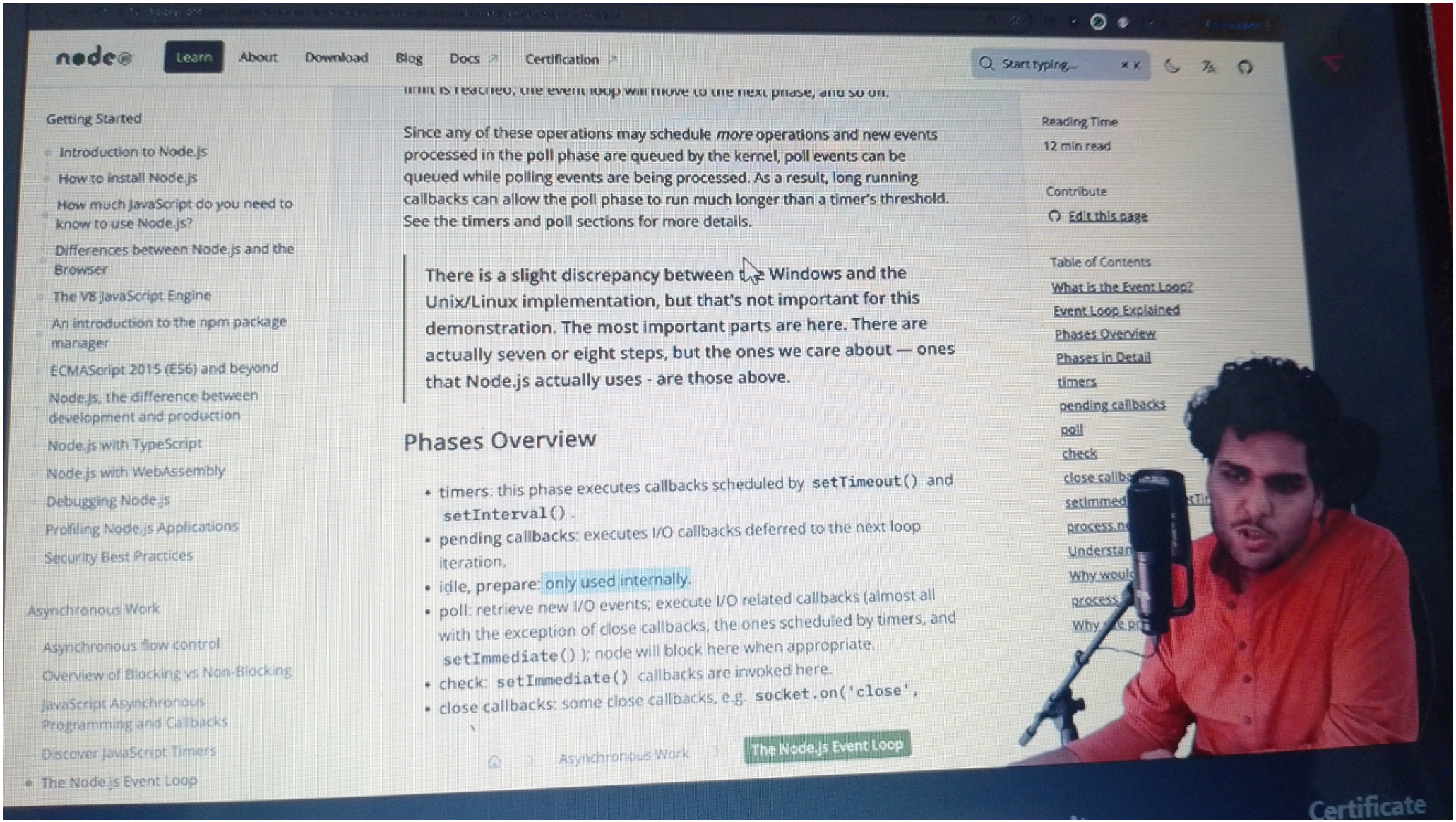
Each phase has a FIFO queue of callbacks to execute. While each phase is special in its own way, generally, when the event loop enters a given phase, it will perform any operations specific to that phase, then execute callbacks in that phase's queue until the queue has been exhausted or the maximum number of callbacks has executed. When the queue has been exhausted or the callback limit is reached, the event loop will move to the next phase, and so on.

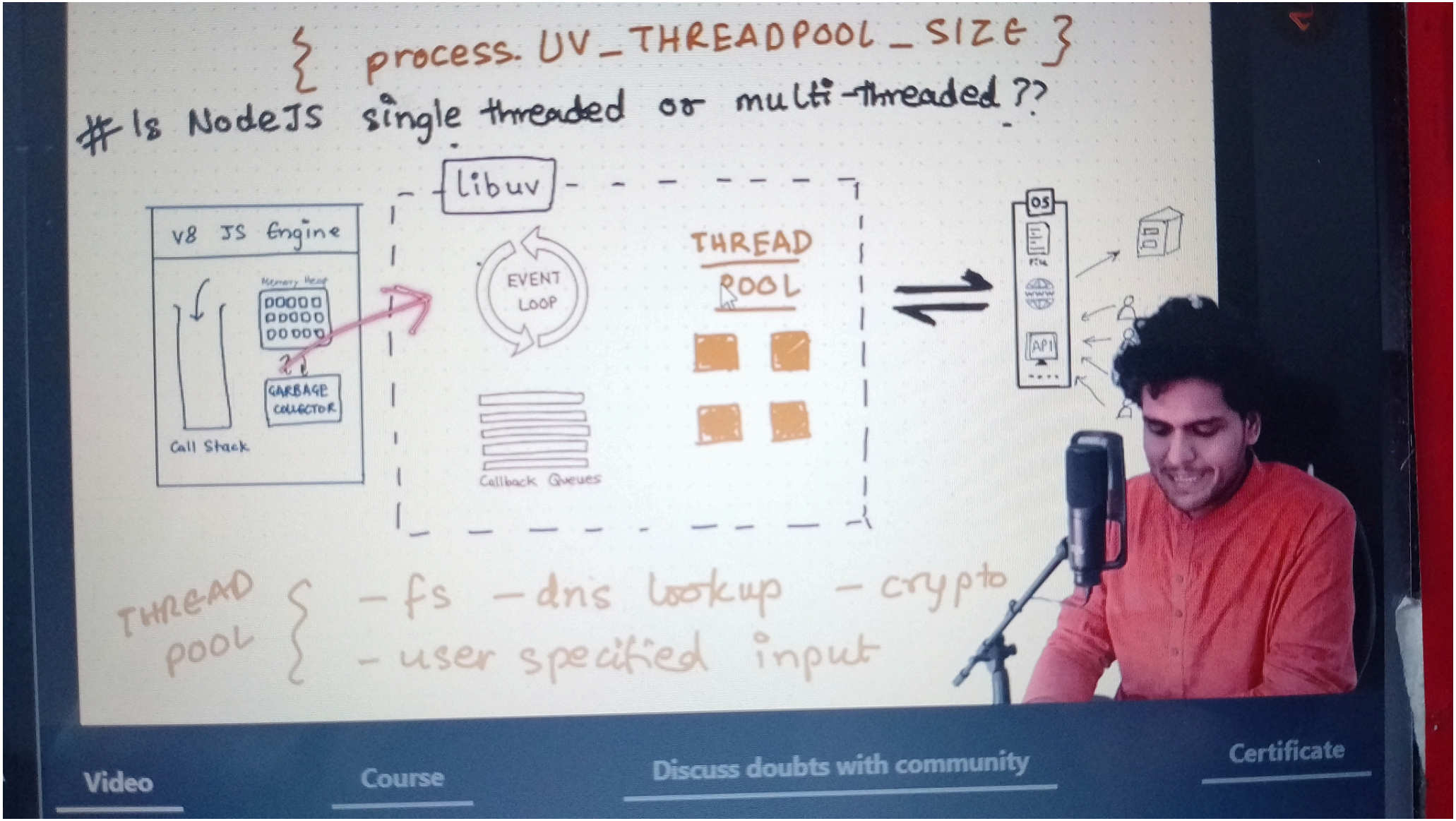
Reading Time
12 min read

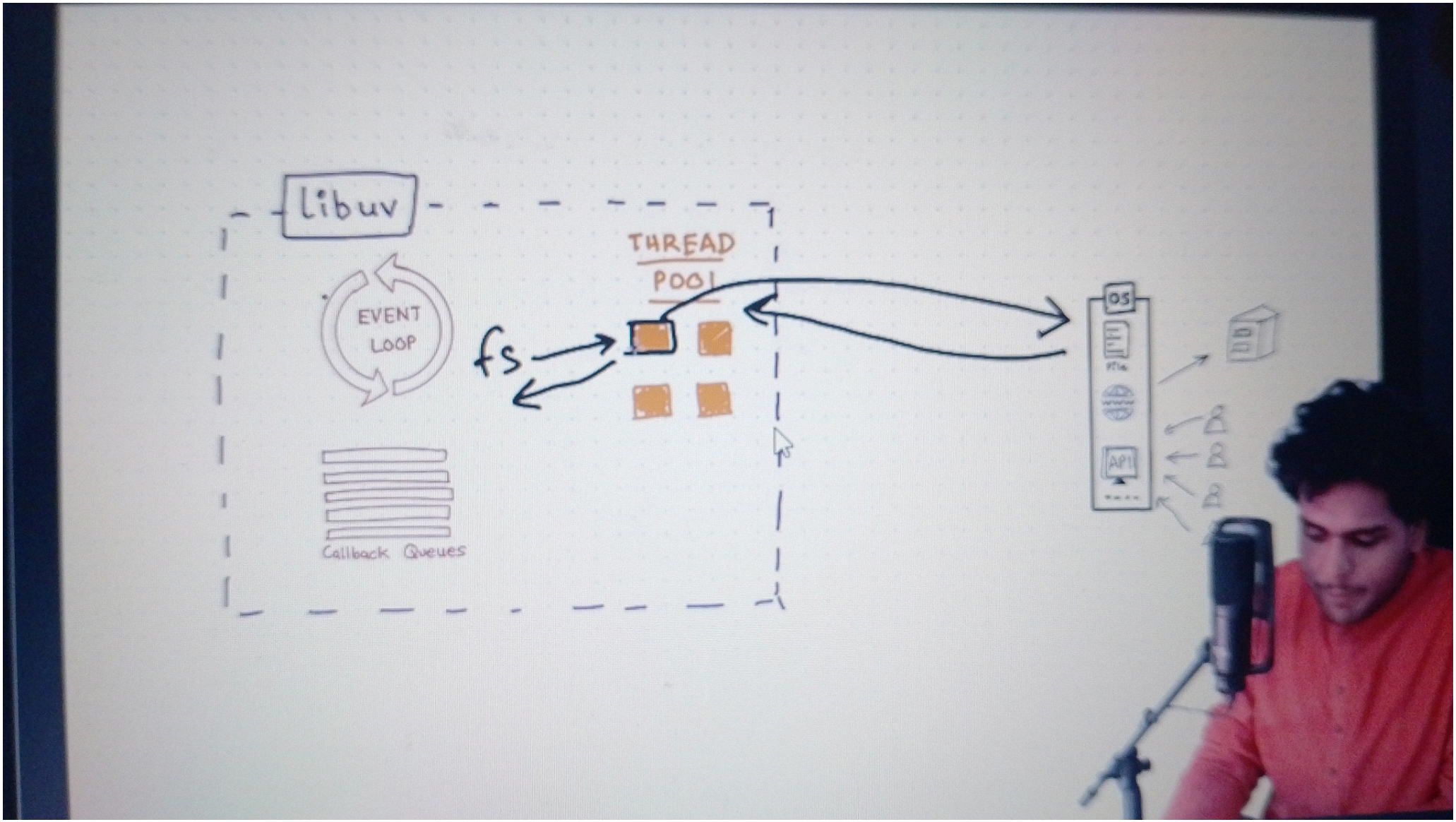
Contribute
Edit this page

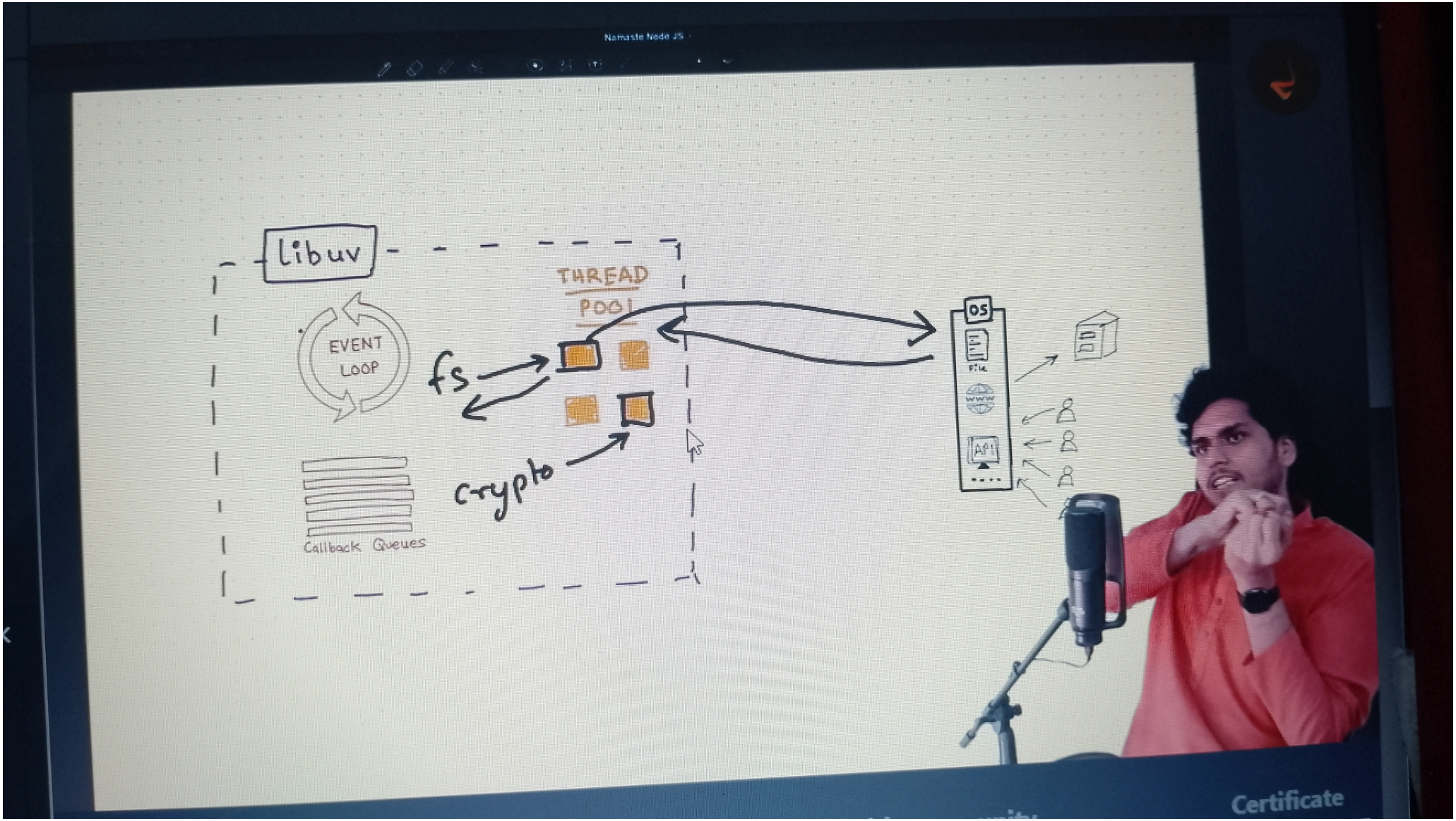
Table of Contents
What is the Event Loop?
Event Loop Explained
Phases Overview
Phases in Detail
timers
pending callbacks
poll
check
close callbacks
setImmediate() vs setTim
process.nextTick()
Understanding process.nex
Why would that be allowed?
process.nextTick(
Why use process.

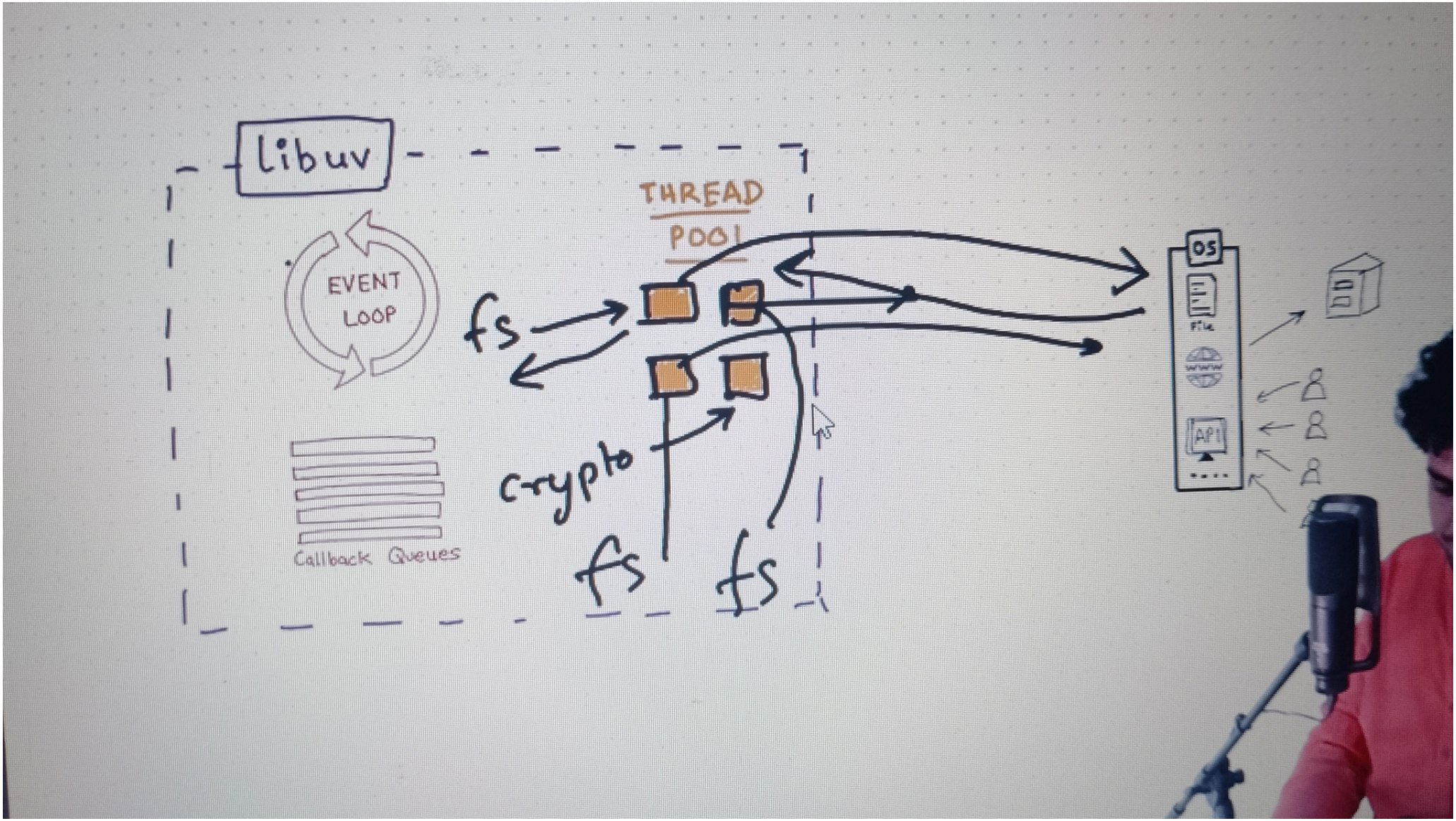

EVENT LOOP
TICK
timer
process.nextTick()
promise callbacks
poll
check
close
V8 JS Engine
nextTick
Promise
Poll
ST
SI
Callback Queues
setImmediate(() => console.log(" 2nd setImmediate"));
setTimeout(() => console.log("Timer expired", 0);
Promise.resolve(() => console.log("Promise"));
fs.readFile("./file.txt", "utf8", () => {
setTimeout(() => console.log("2nd timer"), 0);
process.nextTick(() => console.log("2nd nextTick"));
setImmediate(() => console.log(" 2nd setImmediate"));
console.log("File Reading CB");
process.nextTick(() => console.log("nextTick"));
console.log("Last line of the file.");
console
Last line
nextTick
Promise
Timer expired
setImmediate
File Reading CB
2nd nextTick
2nd setImmediate
2nd timer

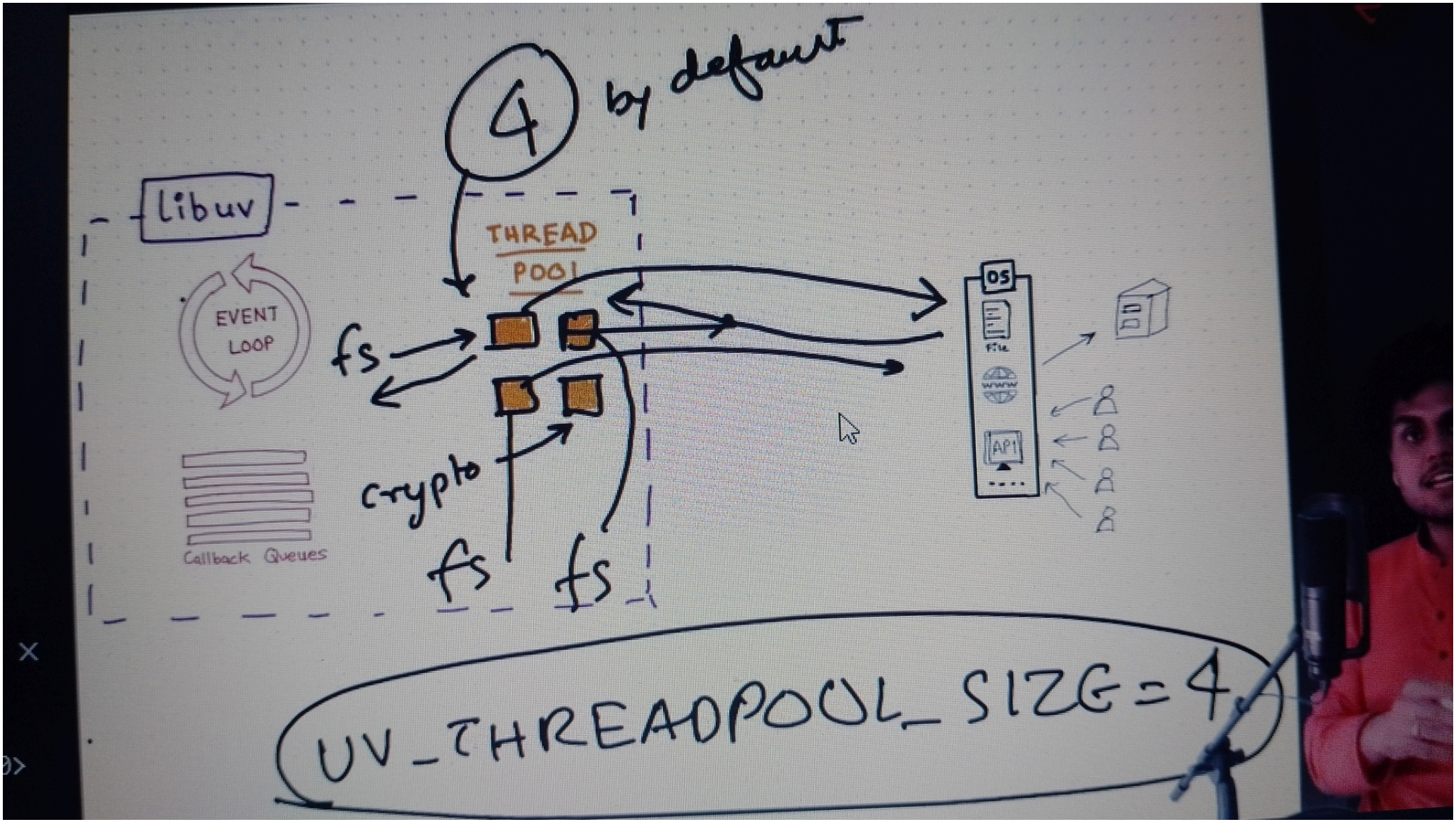

node
Learn
About
Download
Blog
Docs
Certification
Start typing...
limit is reached, the event loop will move to the next phase, and so on.
Since any of these operations may schedule *more* operations and new events processed in the **poll** phase are queued by the kernel, poll events can be queued while polling events are being processed. As a result, long running callbacks can allow the poll phase to run much longer than a timer's threshold. See the **timers** and **poll** sections for more details.
There is a slight discrepancy between the Windows and the Unix/Linux implementation, but that's not important for this demonstration. The most important parts are here. There are actually seven or eight steps, but the ones we care about — ones that Node.js actually uses - are those above.
Phases Overview
timers: this phase executes callbacks scheduled by setTimeout() and setInterval().
pending callbacks: executes I/O callbacks deferred to the next loop iteration.
idle, prepare: only used internally.
poll: retrieve new I/O events; execute I/O related callbacks (almost all with the exception of close callbacks, the ones scheduled by timers, and setImmediate()); node will block here when appropriate.
check: setImmediate() callbacks are invoked here.
close callbacks: some close callbacks, e.g. socket.on('close',
Getting Started
Introduction to Node.js
How to install Node.js
How much JavaScript do you need to know to use Node.js?
Differences between Node.js and the Browser
The V8 JavaScript Engine
An introduction to the npm package manager
ECMAScript 2015 (ES6) and beyond
Node.js, the difference between development and production
Node.js with TypeScript
Node.js with WebAssembly
Debugging Node.js
Profiling Node.js Applications
Security Best Practices
Asynchronous Work
Asynchronous flow control
Overview of Blocking vs Non-Blocking
JavaScript Asynchronous Programming and Callbacks
Discover JavaScript Timers
The Node.js Event Loop
Asynchronous Work
The Node.js Event Loop
Reading Time
12 min read
Contribute
Edit this page
Table of Contents
What is the Event Loop?
Event Loop Explained
Phases Overview
Phases in Detail
timers
pending callbacks
poll
check
Certificate

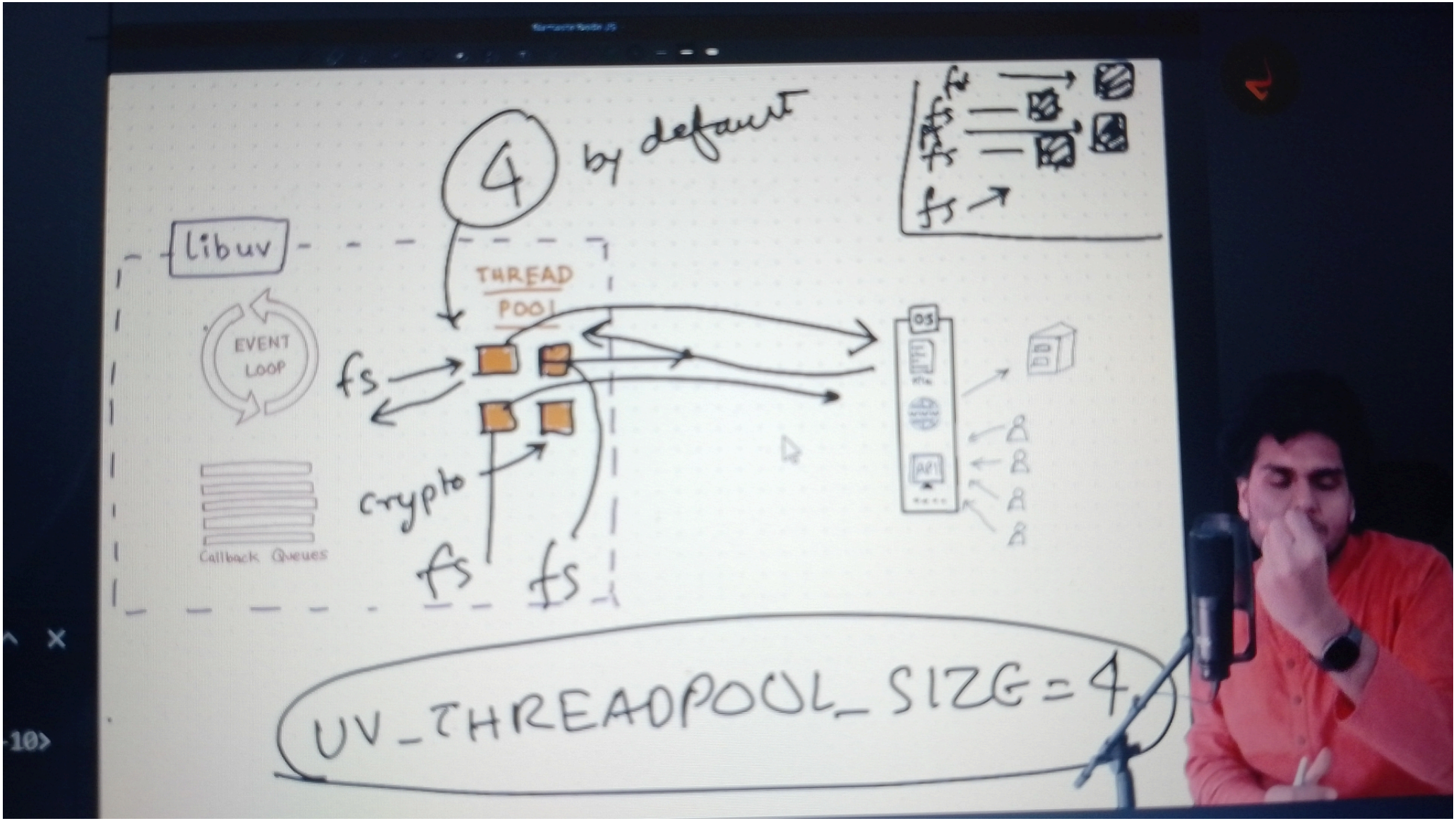

{ process.UV_THREADPOOL_SIZE }
#Is NodeJS single threaded or multi-threaded ??
libuv
V8 JS Engine
Memory Heap
GARBAGE COLLECTOR
Call Stack
EVENT LOOP
Callback Queues
THREAD POOL
OS
API
THREAD POOL
- fs - dns lookup - crypto
- user specified input
Video
Course
Discuss doubts with community
Certificate


libuv
EVENT LOOP
THREAD POOL
fs
Callback Queues
OS
API


Namaste Node JS
libuv
THREAD POOL
EVENT LOOP
fs
crypto
Callback Queues
OS
API
Certificate

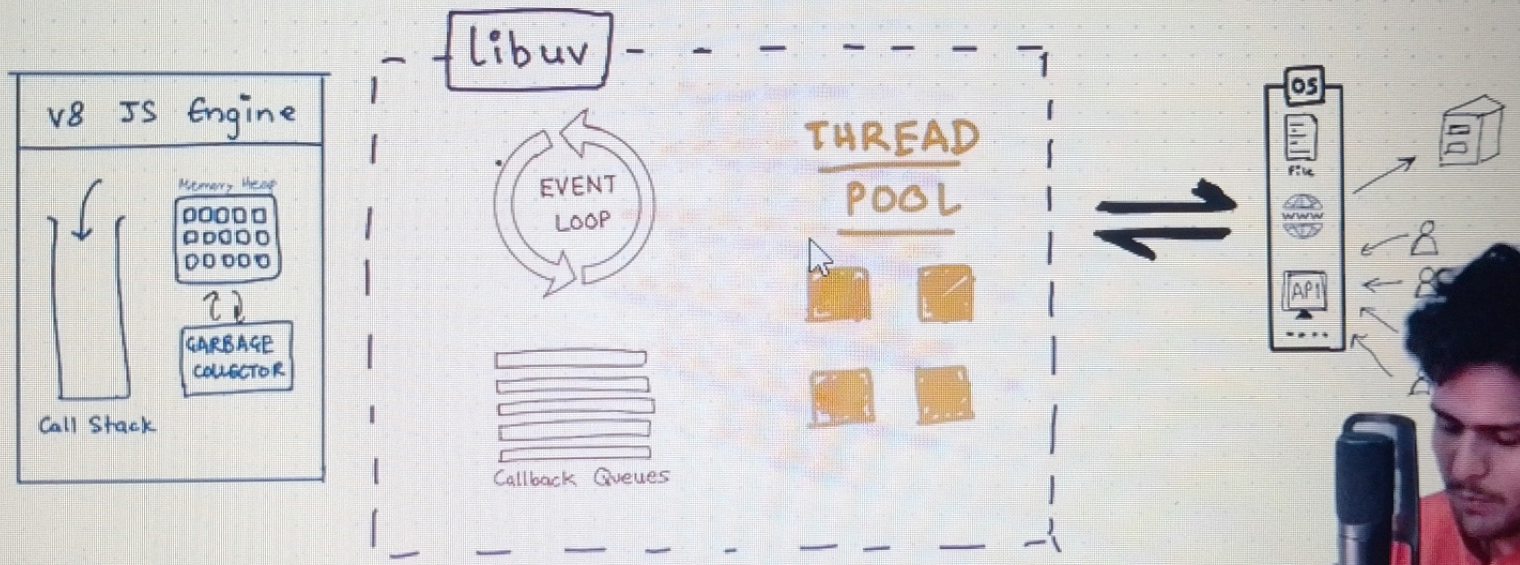

libuv
EVENT LOOP
Callback Queues
THREAD POOL
fs
crypto
fs
fs
OS
API


4
by default
libuv
THREAD POOL
EVENT LOOP
fs
crypto
fs
fs
Callback Queues
OS
File
WWW
API
UV_THREADPOOL_SIZE=4


4
by default
libuv
THREAD POOL
EVENT LOOP
fs
crypto
fs
fs
Callback Queues
OS
API
UV_THREADPOOL_SIZE=4

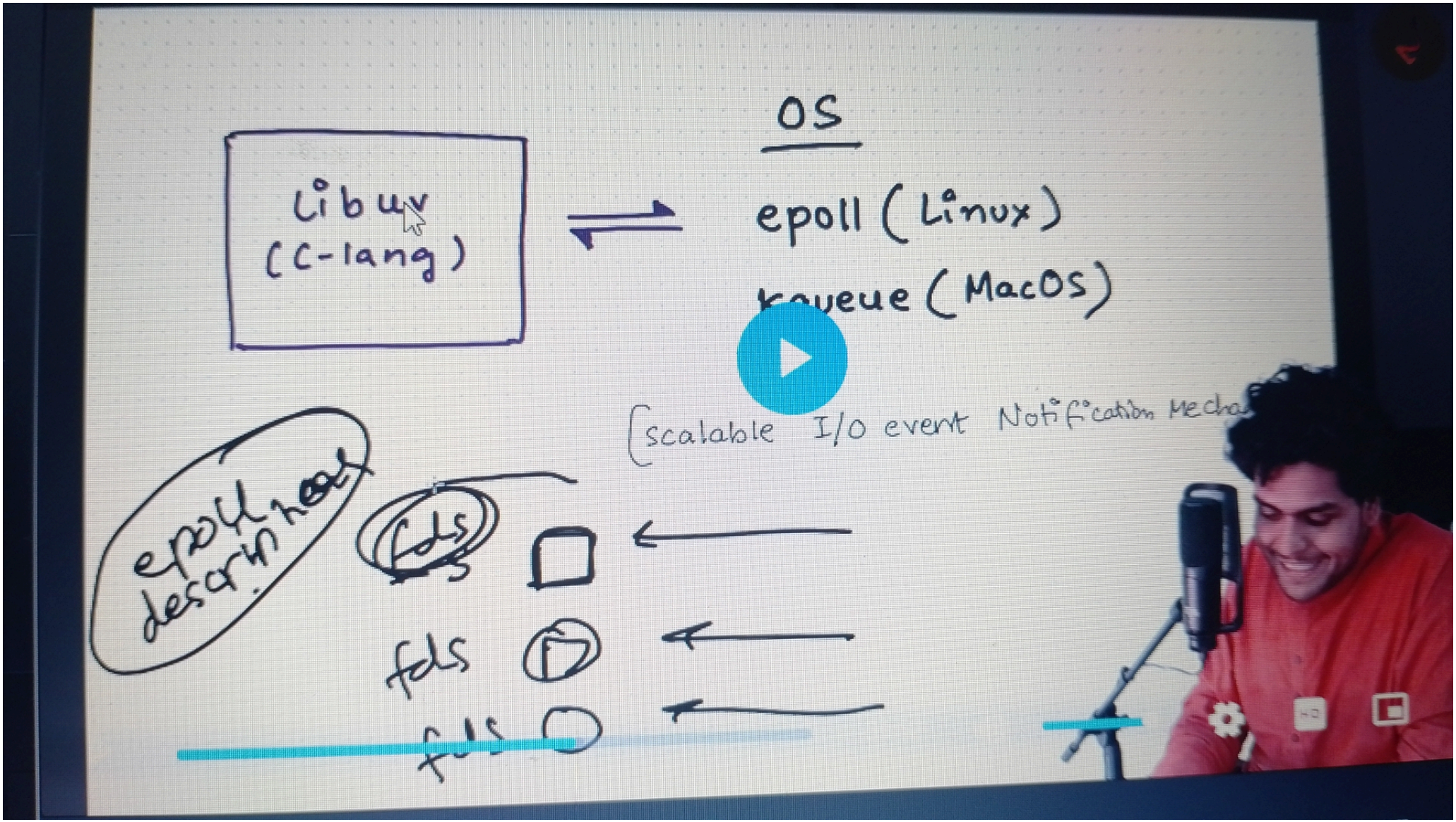
Is NodeJS single threaded or multi-threaded ??

{ process.UV_THREADPOOL_SIZE }



THREAD POOL { - fs - dns.lookup - crypto
- user specified input

Certificate


OS
libuv
(C-lang)
epoll (Linux)
kqueue (MacOS)
(scalable I/O event Notification Mecha
epoll descriptor
fds
fds
fds

NamasteDev.com

NEW

Become Affiliate | Blog | Courses

D

" DON'T BLOCK THE MAIN THREAD "

- sync methods
- complex Regex
- Heavy JSON objects
- Complex calculations/loops

"Data Structures is important"

"Naming is very important"

"There's a lot to learn"

Video | Course | Discuss doubts with community | Certificate

pipes

node thread-po

" DON'T BLOCK THE MAIN THREAD "

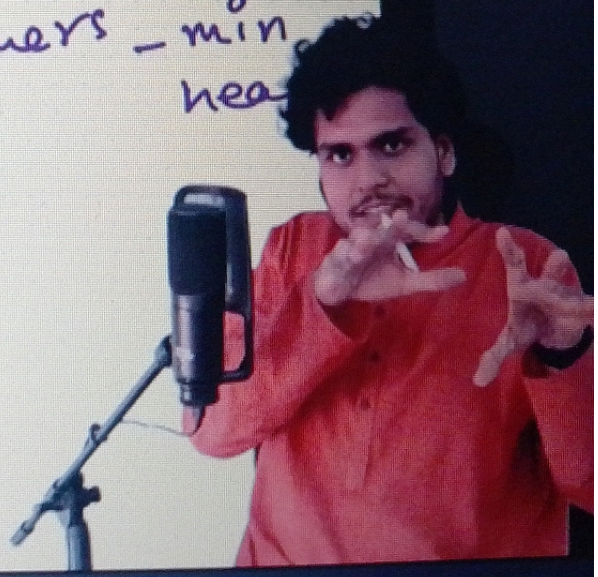
- sync methods
- complex Regex
- Heavy JSON objects
- Complex calculations/loops

"Data Structures is important" epoll-red black tree

timers - min heap

"Naming is very important"

"There's a lot to learn"



aded?
ynchronous co
like crypto,

file descri
is not good
y concurrent
d algorithm
nt notificat
tiple fds
soon as
tifies to li

poll uses, e

e-10> node thread-po

NamasteDev

what is javascript - Googl

namastedev.com/learn/namaste-node/thread-pool-in-...

NamasteDev.com

NEW

Become Affiliate Blog Courses

Namaste Node JS -

" DON'T BLOCK THE MAIN THREAD "

- sync methods
- Heavy JSON objects
- complex Regex
- Complex calculations/loops

"Data Structures is important" epoll-red black tree

timers - min he

"Naming is very important"

procon.nextTick vs setImmediate

"There's a lot to learn"

Video Course Discuss doubts with community Certificate