

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

Node.js

[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

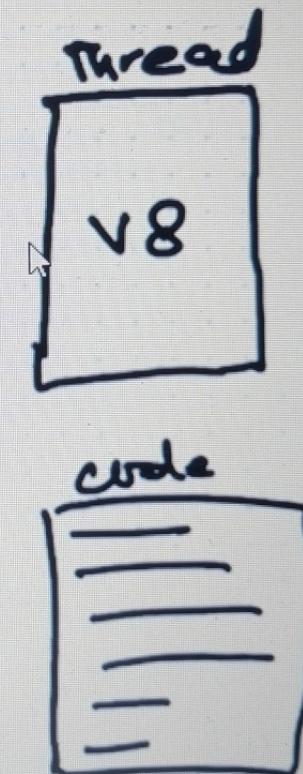
Node.js is a cross-platform, open-source JavaScript runtime environment that can run on Windows, Linux, Unix, macOS, and more. Node.js runs on the V8 JavaScript engine, and executes JavaScript code outside a web browser.

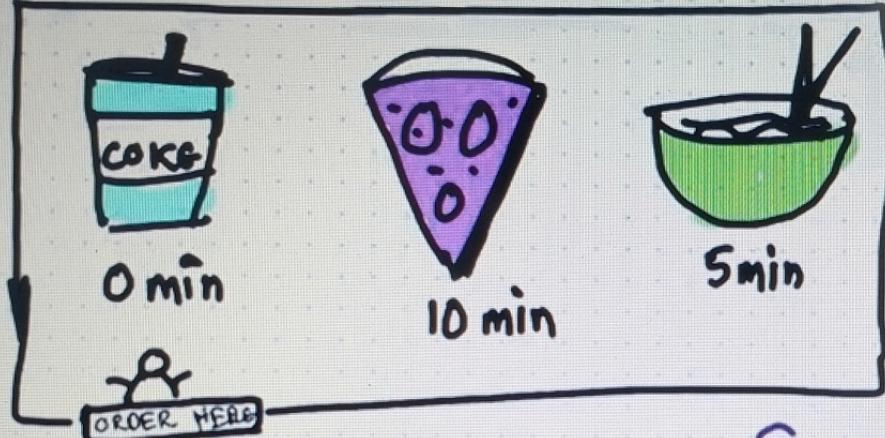
Node.js lets developers use JavaScript to write command line tools and for server-side scripting. The ability to run JavaScript code on the server is often used to generate dynamic web page content before the page is sent to the user's web browser.

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]

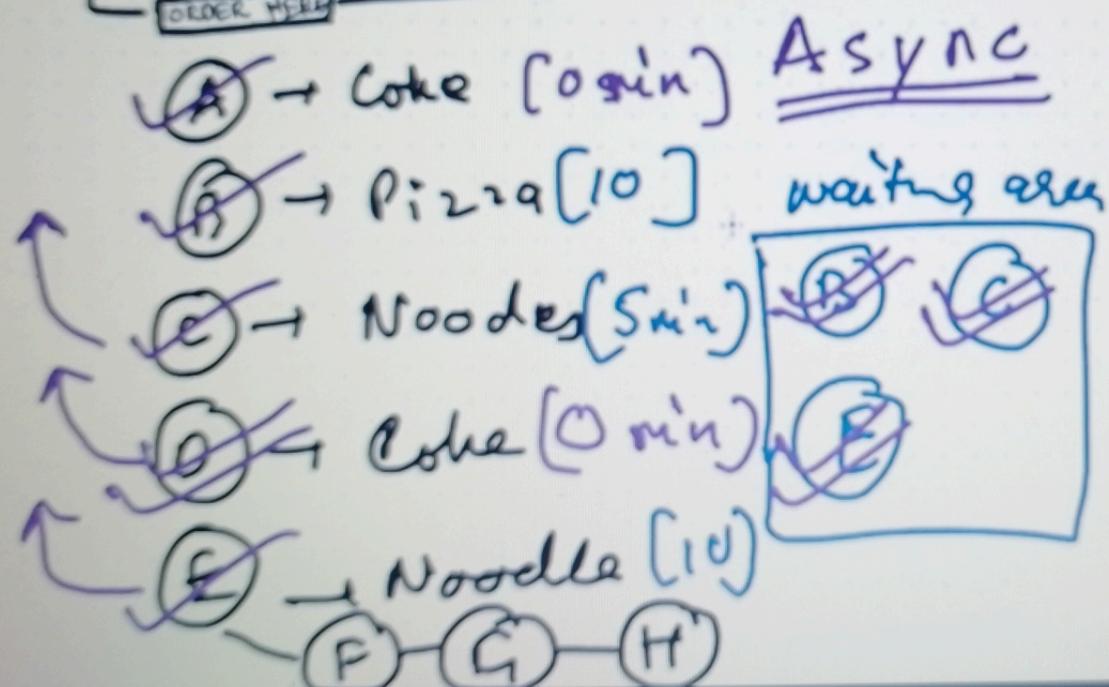
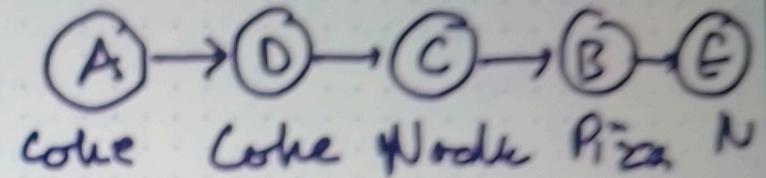
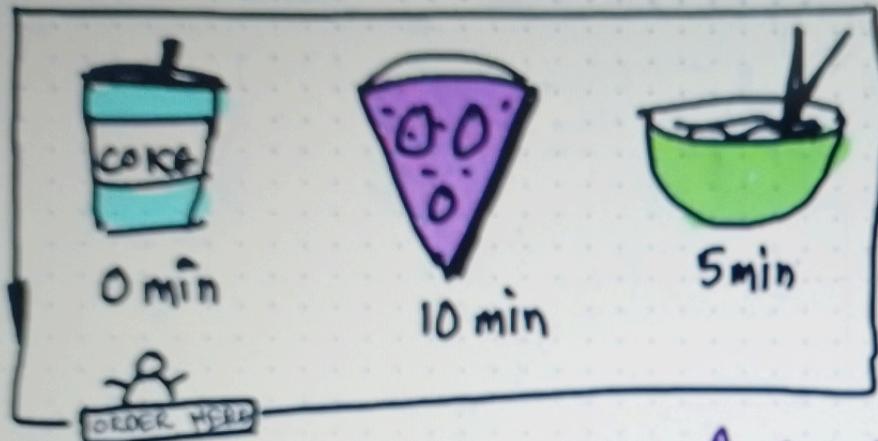
The Node.js distributed development project was previously governed by the Node.js Foundation,^[8] and has now merged with the JS Foundation to form the OpenJS Foundation. OpenJS Foundation is facilitated by the Linux Foundation's Collaborative Projects program.^[9]





- ~~E~~ → Coke [0min] Synchronous
- ~~F~~ → Noodles [5min]
- ~~G~~ → Pizza [15min]
- ~~H~~ → Coke [15min]
- E → Noodles [20min]
- 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 tasks 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);
});

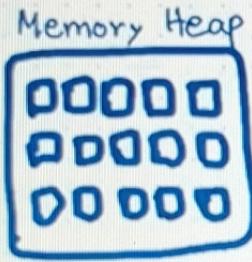
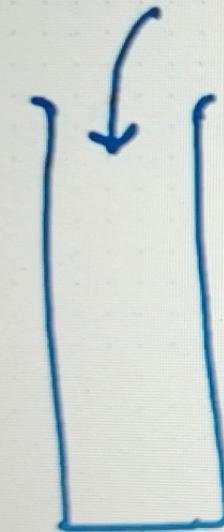
fs.readFile("./gossip.txt", "utf8", (data) => {
  console.log("File Data", data);
});

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

{ These tasks take
time to execute }



v8 JS Engine

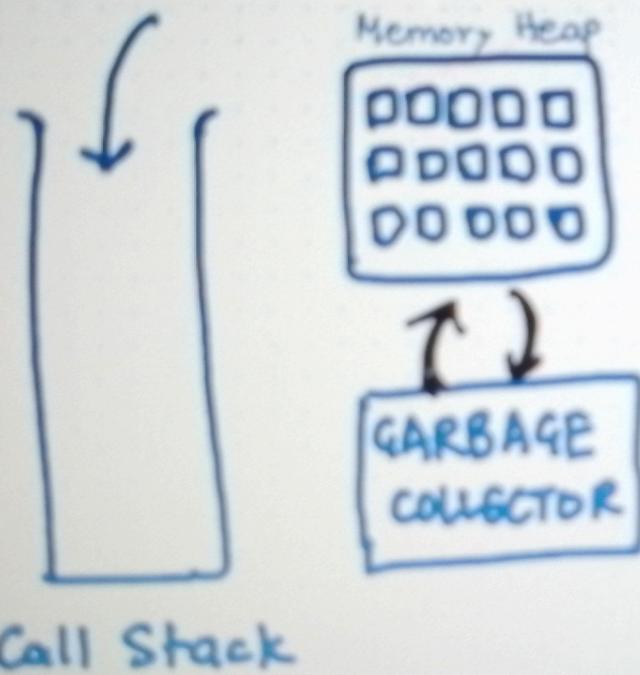


THRASH

```
var a = 1078698;  
var b = 20986;  
  
function multiplyFn(x, y) {  
    const result = a * b;  
    return result;  
}  
  
var c = multiplyFn(a, b);
```

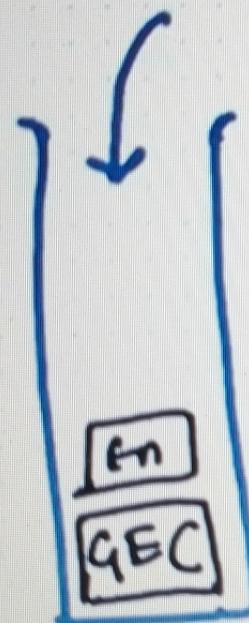


V8 JS Engine

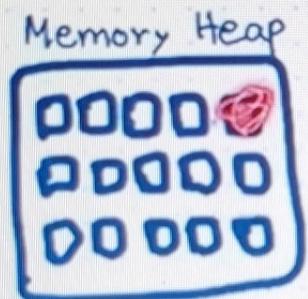


```
var a = 1078698;  
var b = 20986;  
  
function multiplyFn(x, y) {  
    const result = a * b;  
    return result;  
}  
  
var c = multiplyFn(a, b);
```

v8 JS Engine



Call Stack



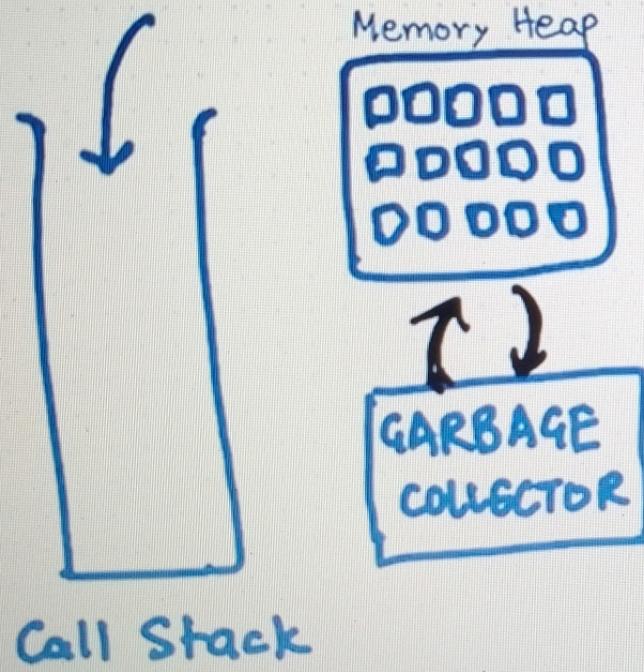
```
var a = 1078698;
var b = 20986;

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

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



v8 JS Engine



```
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 nothing



NamasteDev

namastedev.com/learn/namaste-node/libuv-async-io

Namaste Node JS

6 of 15 Lessons Completed

Season 01 - (Releasing on 09 Aug)

- Episode-00 | Welcome to NamasteNodeJS
- Episode-01 | Introduction to Node.js
- Episode-02 | JS on Server
- Episode-03 | Let's write code
- Episode-04 | module.export & require
- Episode-05 | Diving into the Node.js repo
- Episode-06 | libuv & async I/O
- Episode-07 | sync, async, setTimeout, setInterval

The diagram illustrates the internal structure of the v8 JS Engine. It features a central box labeled "v8 JS Engine" containing a "Memory Heap" (represented by a grid of squares), a "Call Stack" (a vertical list of frames), and a "GARBAGE COLLECTOR". Arrows point from the engine to various external components: "File" (document icon), "DB" (database icon), "WWW" (globe icon), and "Timer" (stopwatch icon).

Video Course Discuss doubts with community Certificate

NamasteDev

namastedev.com/learn/namaste-node/libuv-async-io

Namaste NodeJS

6 of 15 Lessons Completed

Season 01 - (Releasing on 09 August 2024)

- 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 githu
repo
- Episode-06 | libuv & async IO
- Episode-07 | sync, async, setTimeoutZero
code

--NODE JS--

v8 JS Engine

Memory Heap

GARBAGE COLLECTOR

Call Stack

SUPER POWERS

OS

File

DB

WWW

Time

& more.

Video Course Discuss doubts with community Certificate

Episode-06 | libuv & async IO

NamasteDEV

namastedev.com/learn/namaste-node/libuv-async-io

Namaste Node JS

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
- Episode-06 | libuv & async I
- Episode-07 | sync, async, setTi

--NODE JS--

v8 JS Engine

Memory Heap

GARBAGE COLLECTOR

Call Stack

libuv

SUPER HERO

OS

File

DB

www

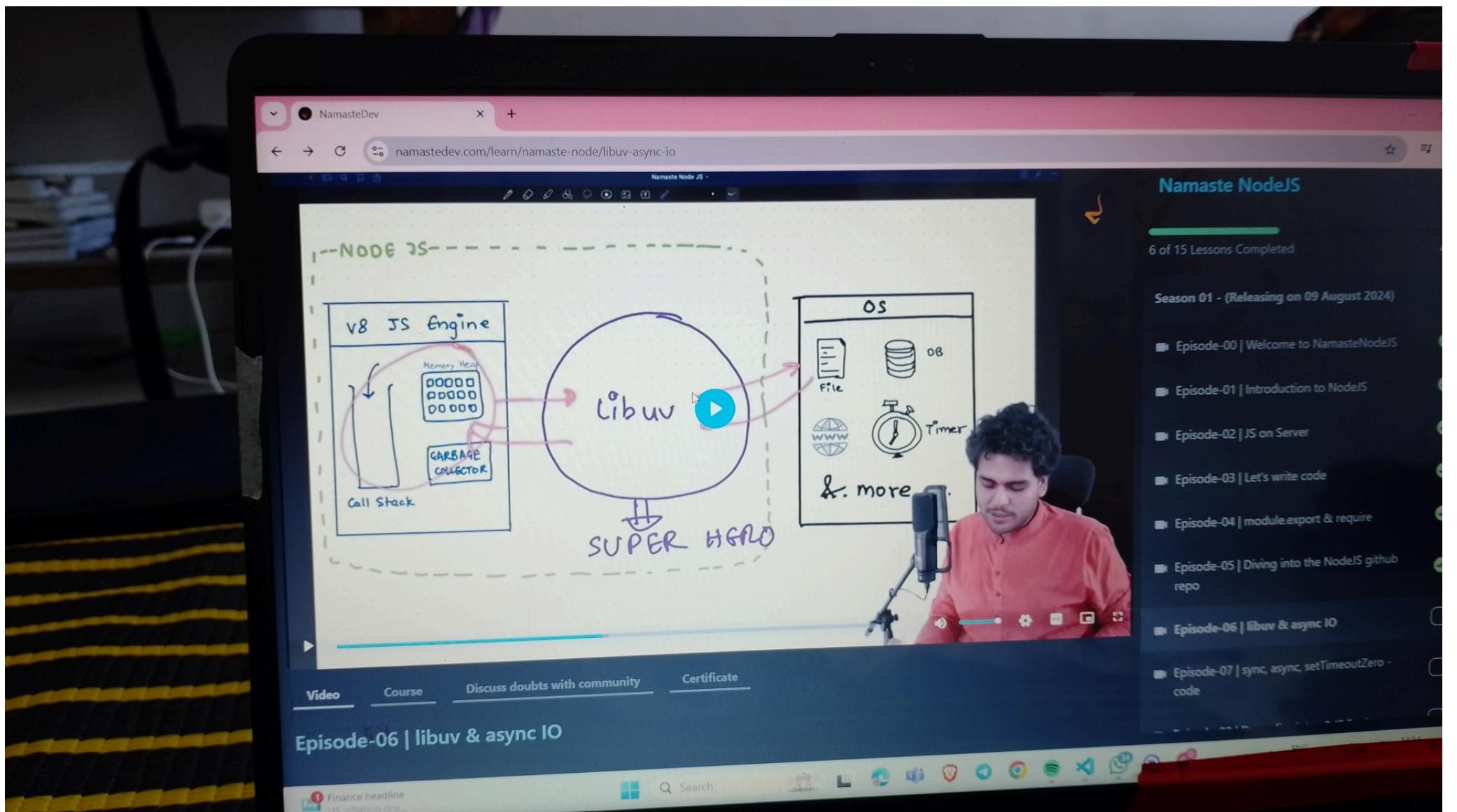
Timer

& more

Video Course Discuss doubts with community Certificate

libuv & async IO

The image shows a video call interface. On the left, there is a hand-drawn diagram of the Node.js architecture. It features a central circle labeled 'libuv'. To its left is a box labeled 'v8 JS Engine' containing a 'Memory Heap' (represented by a grid of squares), a 'GARBAGE COLLECTOR' (represented by a rectangle with arrows), and a 'Call Stack' (represented by a vertical stack of rectangles). A dashed green line separates this from a box labeled 'OS' which contains icons for 'File', 'DB', 'www', and 'Timer'. Below the OS box, the text '& more' is written. To the right of the diagram is a video feed of a man with dark hair, wearing a red shirt, sitting in front of a microphone. He is gesturing with his hands while speaking. The video player has a play button and a progress bar at the bottom. At the very bottom of the screen, there are navigation links for 'Video', 'Course', 'Discuss doubts with community', and 'Certificate'.



NamasteDev

namastedev.com/learn/namaste-node/libuv-async-io

Namaste Node JS

--NODE JS--

V8 JS Engine

Memory Heap

Call Stack

GARBAGE COLLECTOR

offloads

libuv

GENIE

SUPER HERO

& more

OS

File

DB

www

Timer

Episode-00 | Welcome to Namaste

Episode-01 | Introduction to Node.js

Episode-02 | JS on Server

Episode-03 | Let's write code

Episode-04 | module.export & require

Episode-05 | Diving into the Node.js repo

Episode-06 | libuv & async IO

Episode-07 | sync, async, setTimeout code

Episode-08 | Deep dive into v8 JS Engine

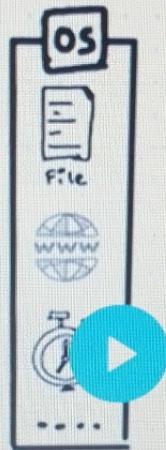
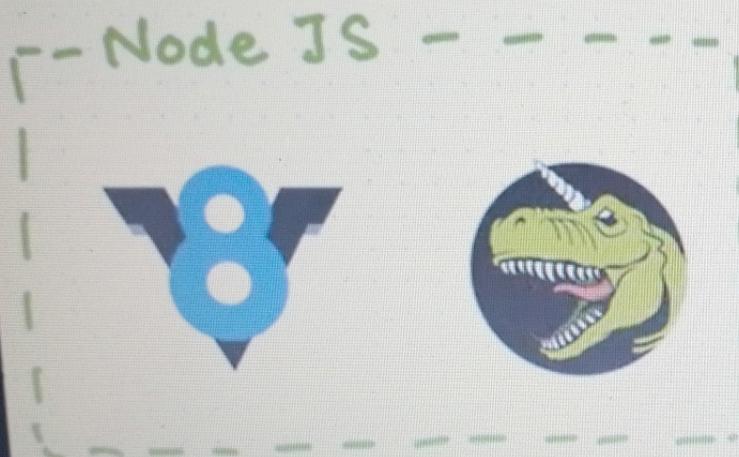
Episode-09 | libuv & Event Loop

Episode-10 | Thread pool in libuv

Episode-11 | Creating a Server

Video Course Discuss doubts with community Certificate

Episode-06 | libuv & async IO

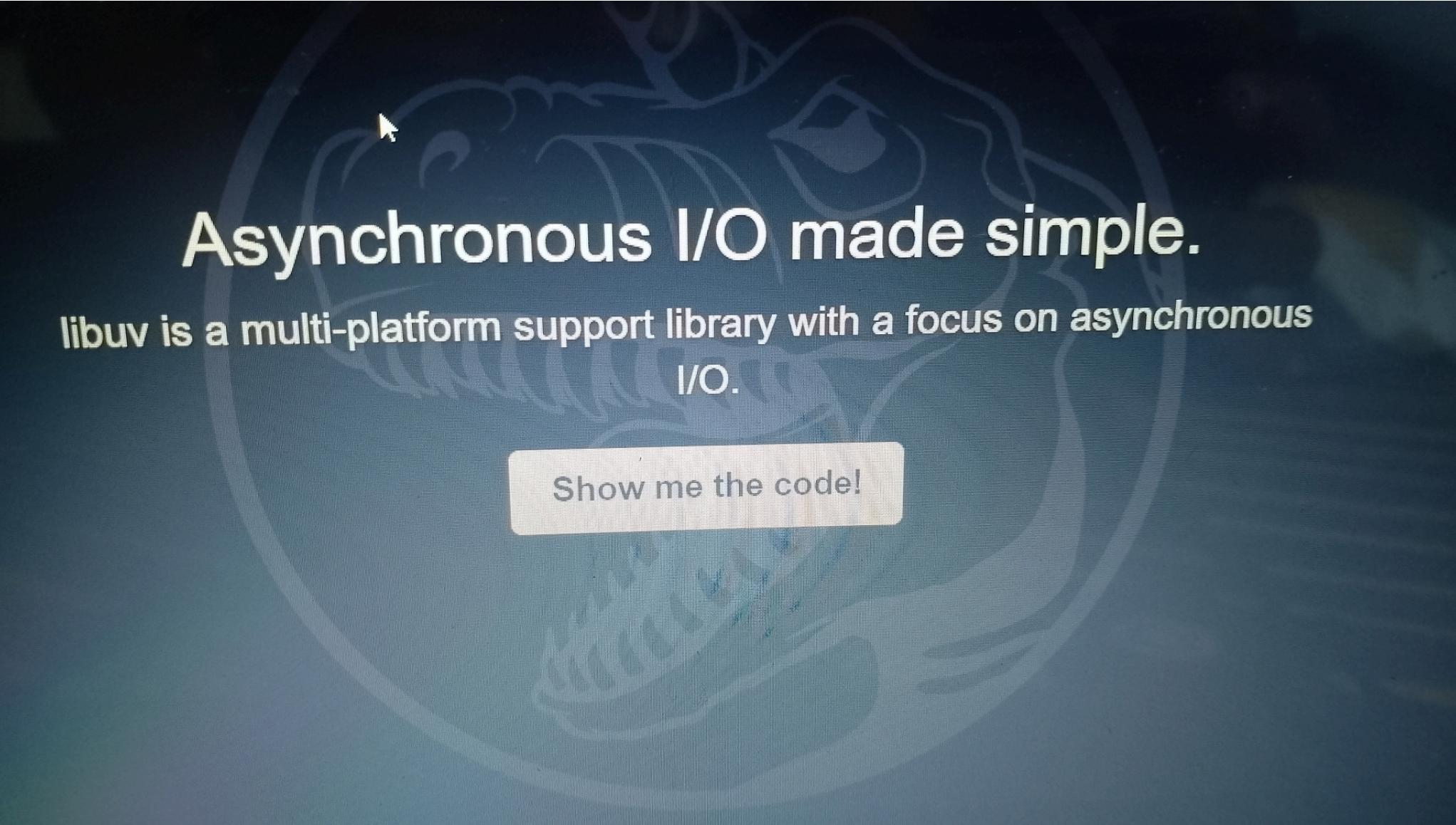


```
https.get("https://api.fbi.com", (res) => {
  console.log("secret data:" + res.secret);
});

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

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





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 --

V8 C++ C

Async I/O made simple

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);
```

E

Ep

Ep

Ep

Ep

Ep

Ep

Ep

Ep

namastedev.com/learn/namaste-node/libuv-async-io

The screenshot shows a video player interface for a Node.js tutorial. On the left, there's a hand-drawn style diagram of the V8 JS Engine. It features a blue '8' icon, a 'Memory Heap' grid, and a 'GARBAGE COLLECTOR'. To the right of the engine is a green 'libuv' logo with a yellow 'u' and a green T-Rex head. Further right is a terminal window displaying Node.js code. The code includes variable assignments, an HTTPS request, a setTimeout function, an fs.readFile operation, a multiplication function, and a final console.log statement. On the far right, a man with dark hair and a beard, wearing a red shirt, is seated at a desk with a microphone, looking down at his laptop. A progress bar at the bottom indicates the video is at 30% completion. The top of the screen shows the URL 'namastedev.com/learn/namaste-node/libuv-async-io'. A sidebar on the right lists 13 episodes:

- Episode-03 | Let's write a module
- Episode-04 | module exports
- Episode-05 | Diving into libuv repo
- Episode-06 | libuv & Event loop
- Episode-07 | sync, async code
- Episode-08 | Deep dive into libuv
- Episode-09 | libuv & Express
- Episode-10 | Thread pool
- Episode-11 | Creating a database
- Episode-12 | Databases
- Episode-13 | Creating a mongoDB

Namaste Node JS -

-- Node JS --

libuv

V8 JS Engine

Memory Manager
Call Stack
GARbage COLLECTOR

API (A)

SetTimeout (D)

fs (C)

OS File www ...

A

B

C

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;
}

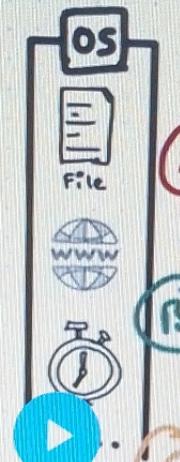
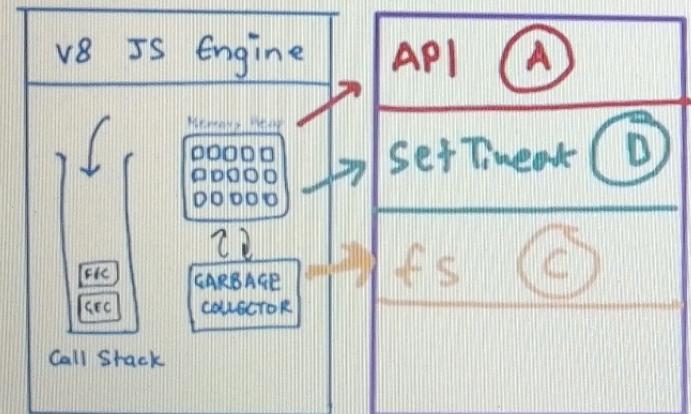
var c = multiplyFn(a, b);

console.log(c);
```

20%

Video Course Discuss doubts with community Certificate

-- Node JS --



```
var a = 10/8698;
var b = 20986;

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

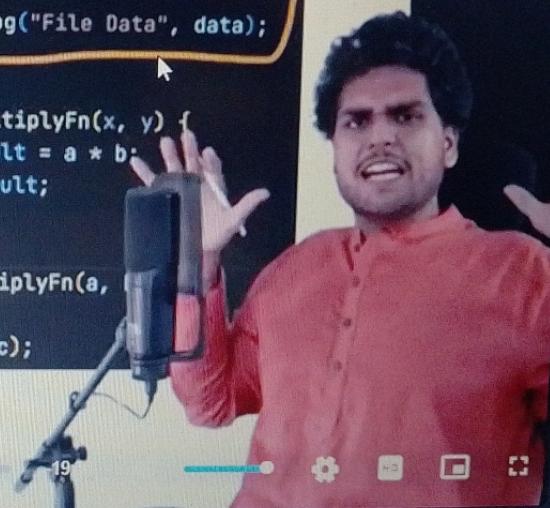
setTimeout(() => {
  console.log("setTimeout");
}, 5000);

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

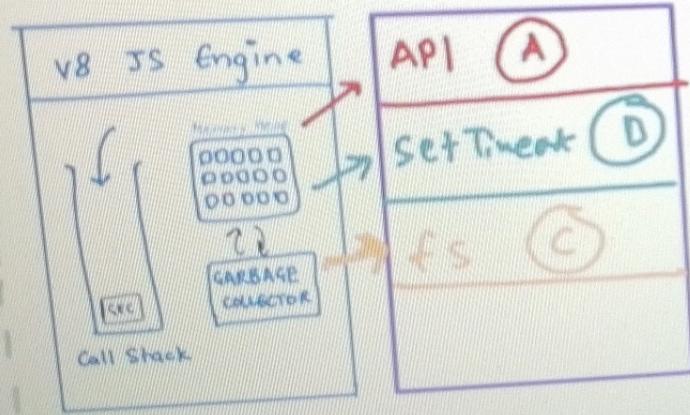
function multiplyFn(x, y) {
  const result = a * b;
  return result;
}

var c = multiplyFn(a,
  b);

console.log(c);
```



-- Node JS --



The screenshot shows a video player interface. On the right, a man in a red shirt is speaking into a microphone. On the left, a code editor displays the following Node.js script:

```
var a = 1078698;
var b = 28986;

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);
```

The code is annotated with circled letters A, B, C, and D. A red circle highlights the first line of the https.get block. A green circle highlights the setTimeout block. An orange circle highlights the fs.readFile block. A blue circle highlights the self-defined multiplyFn function.

- Episode-01
- Episode-04
- Episode-05 | repo
- Episode-06 |
- Episode-07 | sys code
- Episode-08 | De
- Episode-09 | libu
- Episode-10 | Thre
- Episode-11 | Creat
- Episode-12 | Datab
- Episode-13 | Creatin mongodb

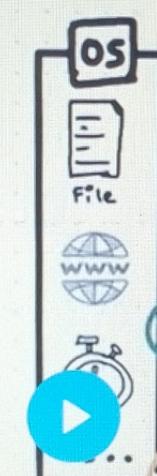
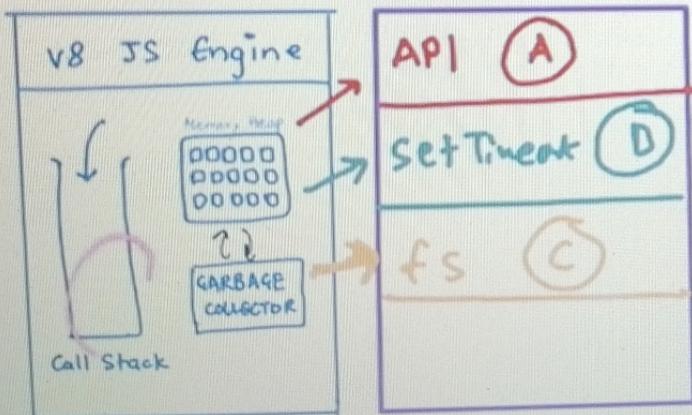
Video

Course

Discuss doubts with community

Certificate

-- Node JS --



```
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);

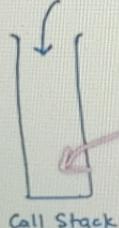
console.log(c);
```



-- Node JS - - - -



v8 JS Engine



API

SetTimeout (A) (D)

FS (C) (E)



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

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

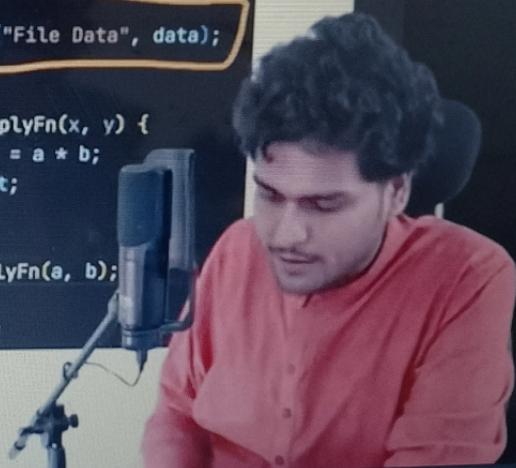
setTimeout(() => {
  console.log("setTimeout");
}, 5000);

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

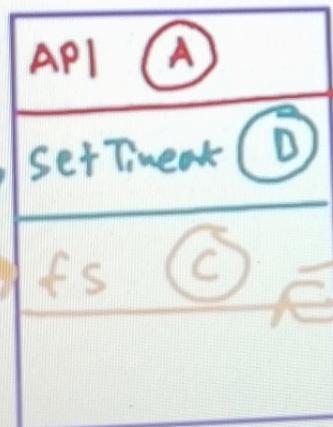
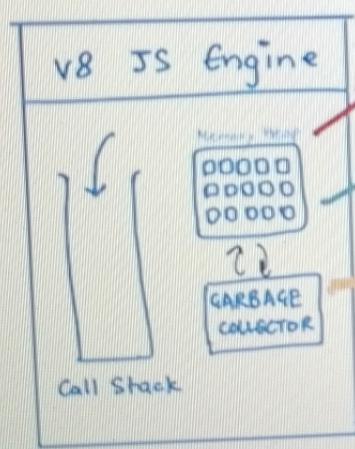
function multiplyFn(x, y) {
  const result = a * b;
  return result;
}

var c = multiplyFn(a, b);

console.log(c);
```



-- Node JS --



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

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

setTimeout(() => {
  console.log("setTimeout");
}, 5000);

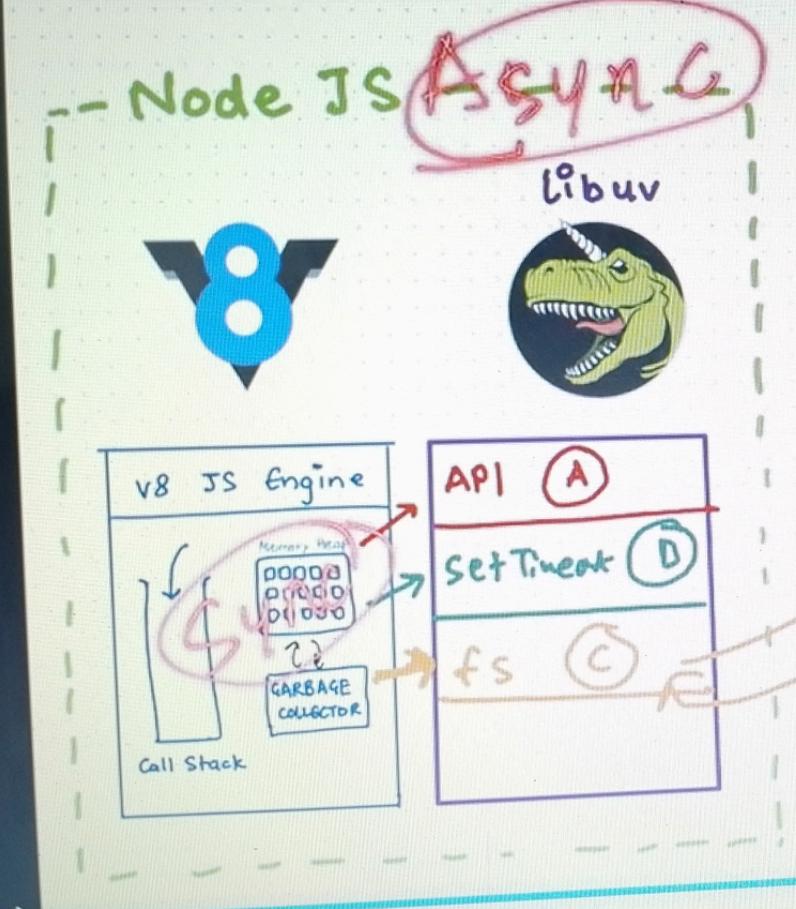
fs.readFile("./gossip.txt", "utf8",
(data) => {
  console.log("File Data", data);
});

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

var c = multiplyFn(a, b);

console.log(c);
```





```

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);
  
```



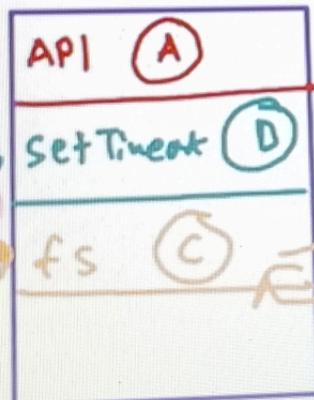
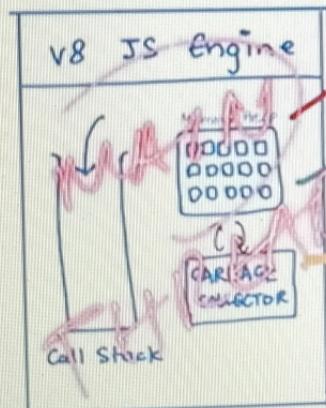
Video

Course

Discuss doubts with community

Certificate

-- Node JS --



```
var a = 1878698;
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)