



SLIDE



Node JS



Node JS introduction & NPM



Introduction To Node.js and Package Manager

4



Introduction To

Node.js and Package Manager





1995-2009 April





4



Javascript was created as client side language .
it could only handle the frontend logics



Javascript Could Not do:

-  Query in database
-  Handle server side request and response
-  Read/Write file on server
-  Server side operations



Ryan Dahl
In 2009 May

created a magical thing
which now we are calling
node.js

4



4



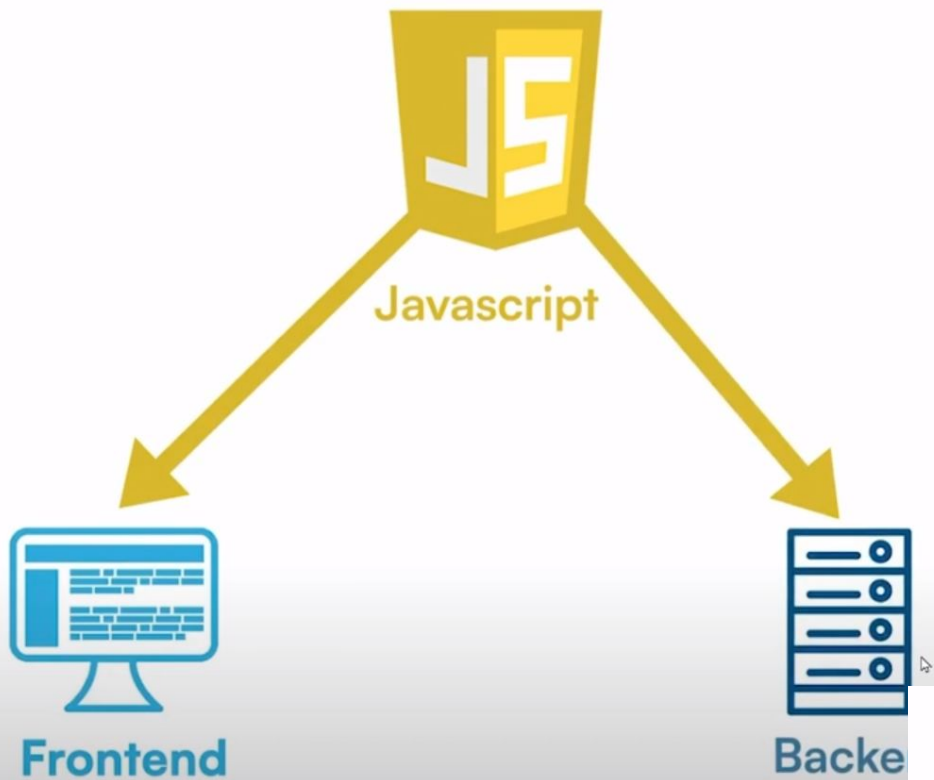
0:55 / 3:13



4



1:01 / 3:13





Now Javascript can do



Query in database



Handle server side request and response



Read/Write file on server



Server side operations



So what is node.js ?



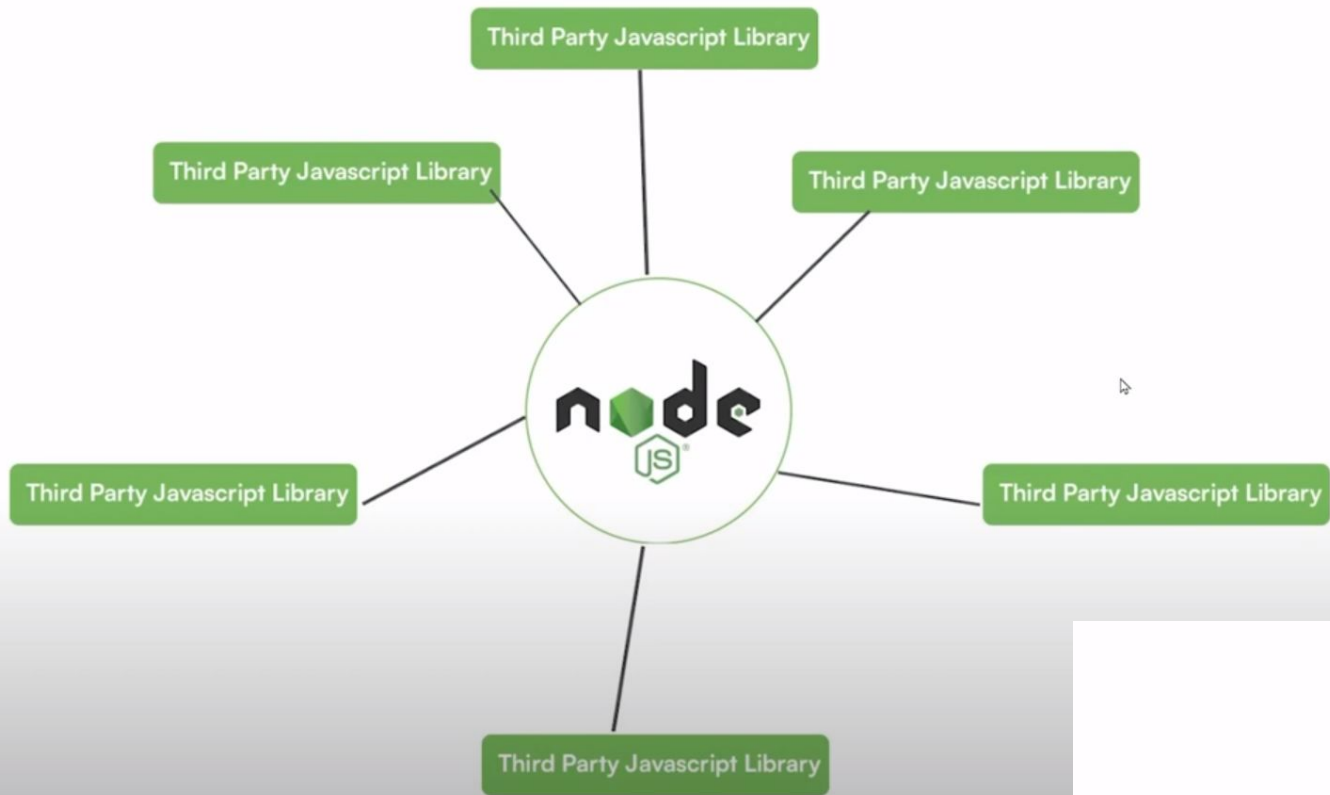
Is a server side javascript runtime



Not a Programming Language **X**



Has Own Library For Manage Node.js





Third Party Libraries

What is NodeJS?



Node JS -

- Open source
- Cross platform
- JS runtime environment
- Allows server side scripting
- Single threaded, Non-blocking
- Capable of asynchronous I/O
- Has event-driven architecture

Runtime environment



Different data values and functions are available, and these differences help distinguish front-end applications from back-end applications.

- **Front-end JavaScript** applications are executed in a **browser's runtime** environment and have access to the **window object**.
- **Back-end JavaScript** applications are executed in the **Node runtime** environment and have access to the **file system, databases, and networks attached to the server**.

JS code may be executed in one of **two runtime environments**:

- A browser's runtime environment
- The Node runtime environment

Node JS Runtime environment

JavaScript Engine



Execution Context Stack

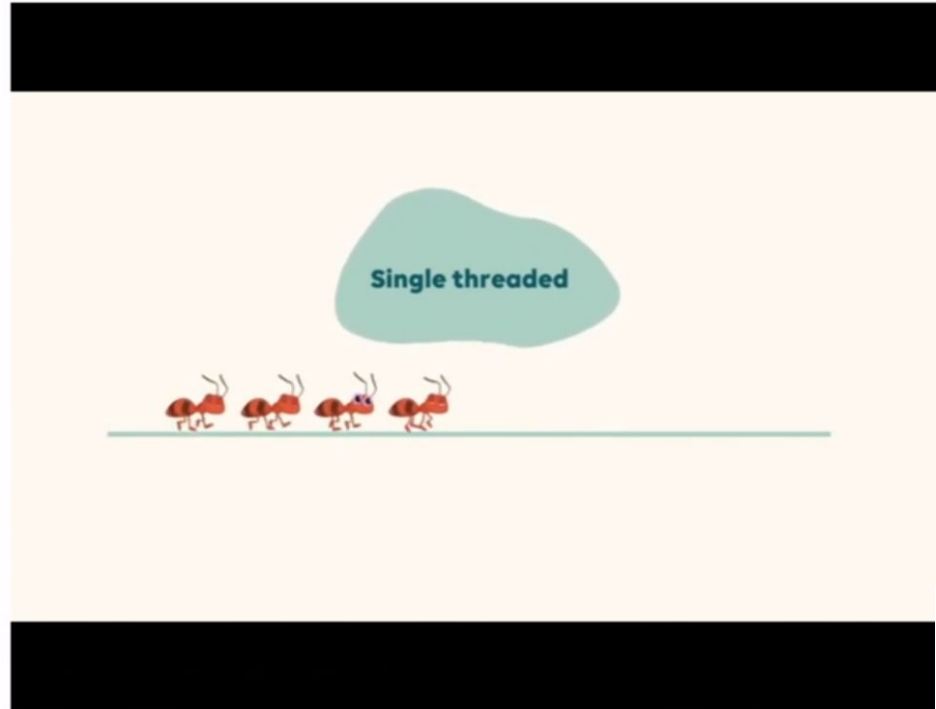
Web API



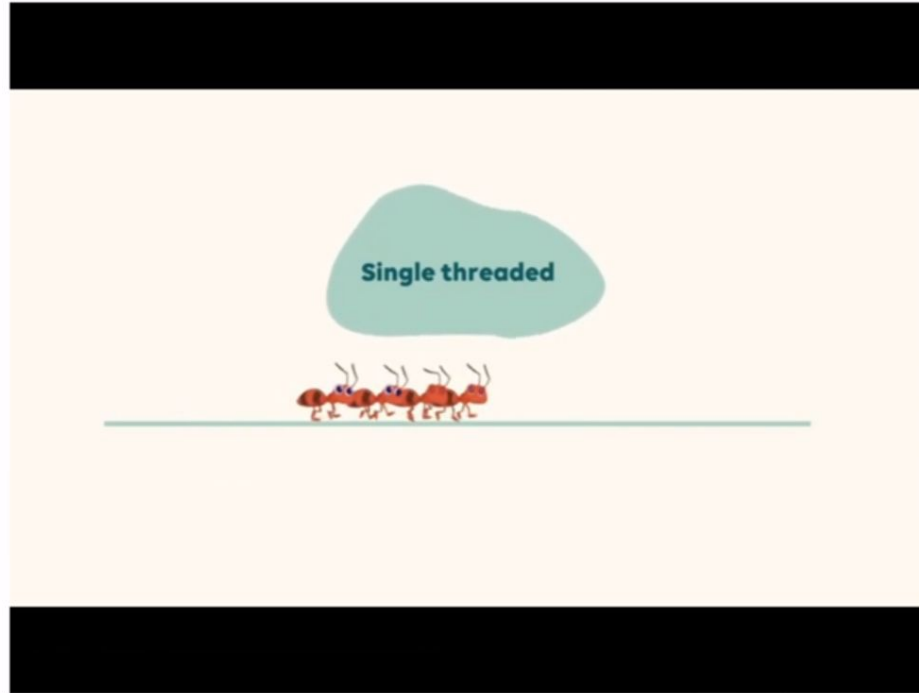
Callback Queue

In 2009, the Node runtime environment was created for the purpose of **executing JavaScript in the browser**, thus enabling programmers to create full-stack (front-end and back-end) JavaScript language.

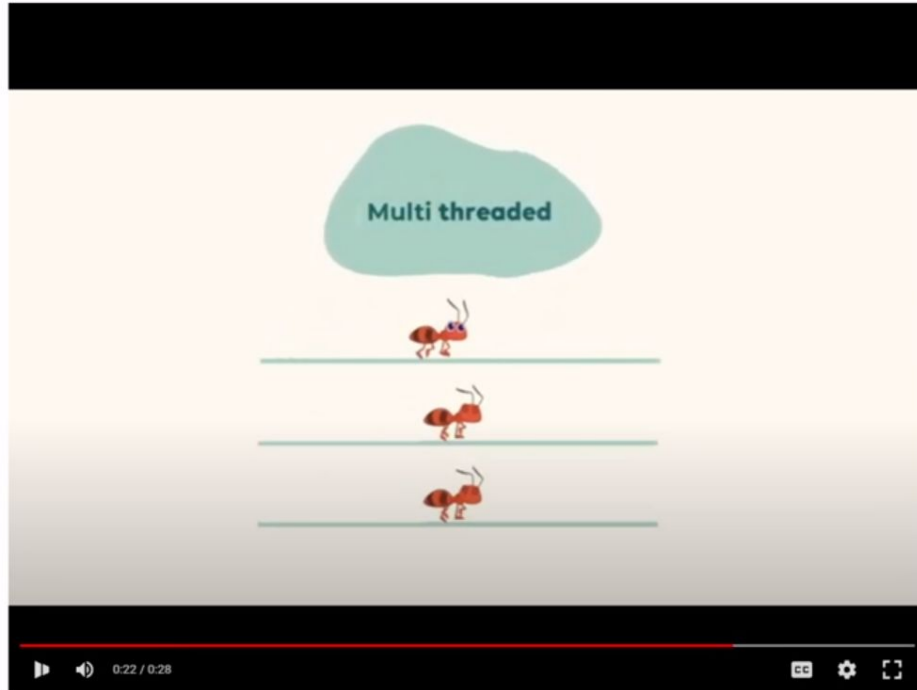
Single thread, Multi thread



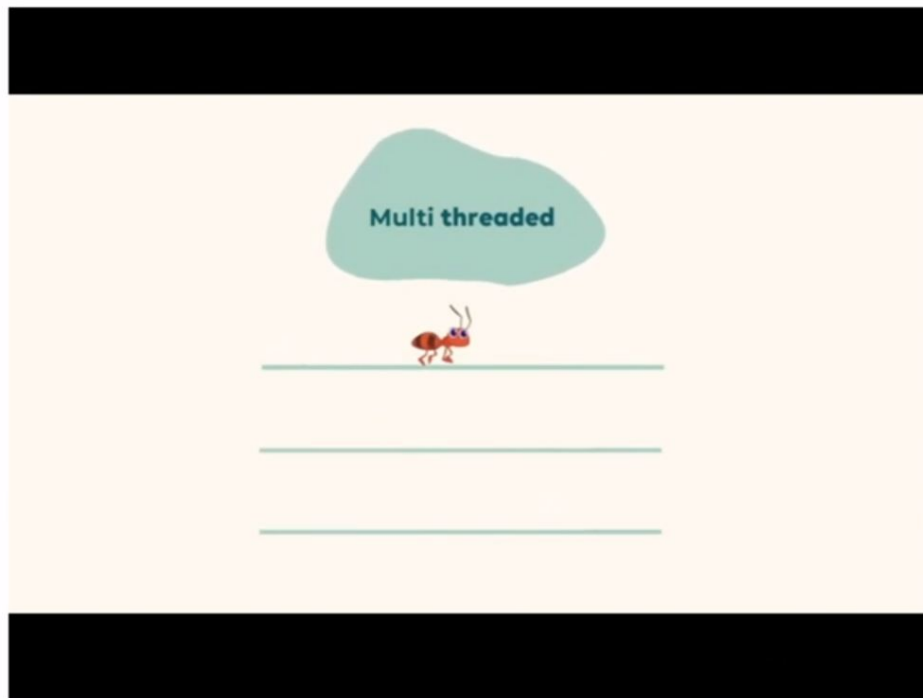
Single thread, Multi thread



Single thread, Multi thread

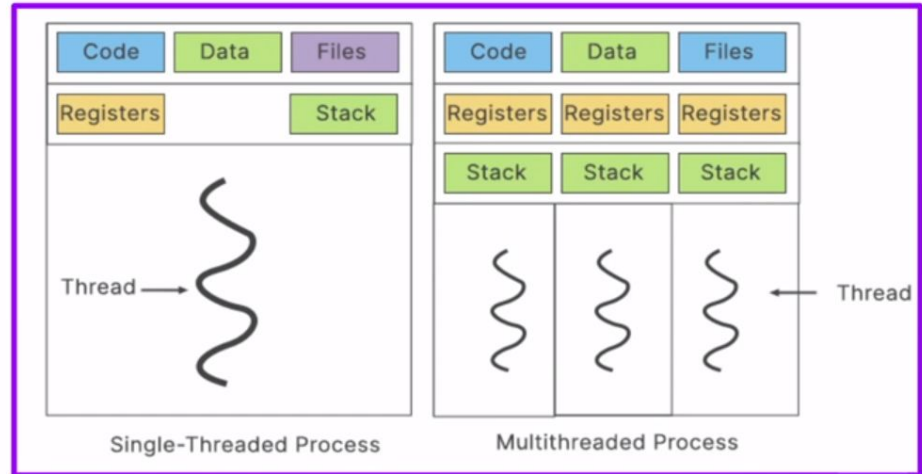


Single thread, Multi thread

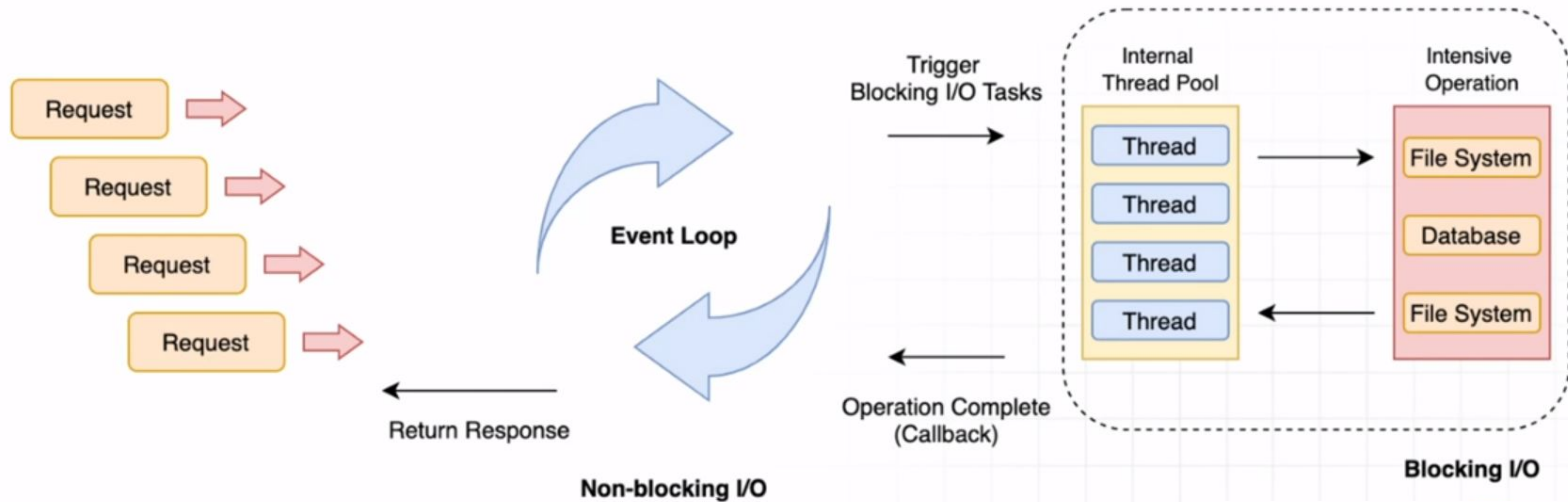


What is thread? Single thread, Multi thread

- ★ A **basic unit of CPU utilization**.
- ★ A thread is a **path of execution** within a process/ task.

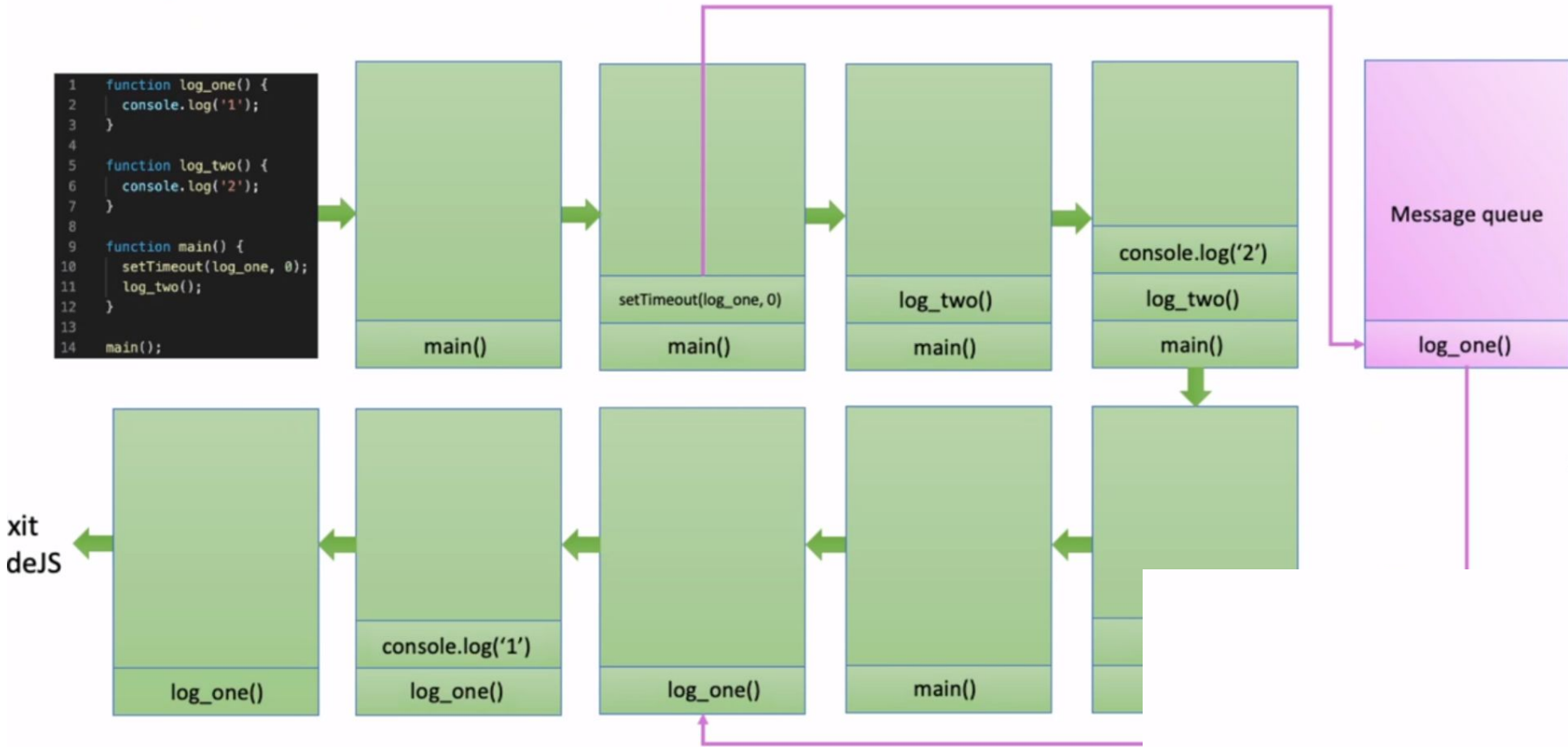


Non-blocking I/O event loop



Asynchronous NodeJS

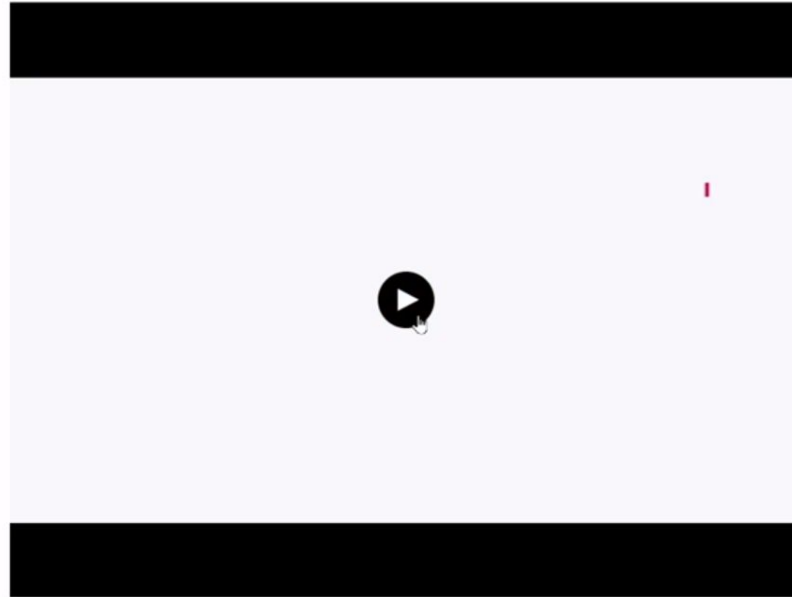
```
1 function log_one() {
2   | console.log('1');
3 }
4
5 function log_two() {
6   | console.log('2');
7 }
8
9 function main() {
10  | setTimeout(log_one, 0);
11  | log_two();
12 }
13
14 main();
```



Asynchronous NodeJS

Further to the **message queue**, the **job queue** was put in place since ES6/ES2015 which is **used by Promises and async/await**. Unlike the message queue, when a promise resolves before the current function execution ends, the callback logic will be executed right after the current function call. This means **callbacks put on the job queue will execute as soon as possible and not deferred till after the call stack is empty**.

Event-driven architecture/ application

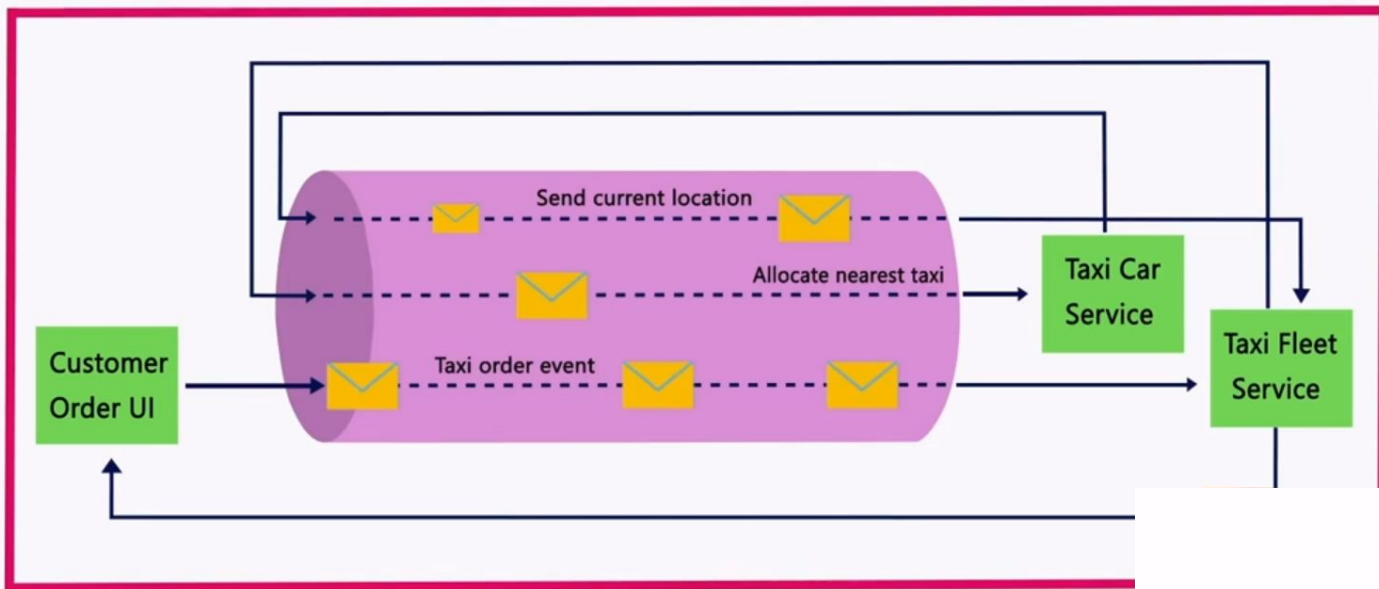


An event-driven application is a program that is written to respond to actions generated by the user or the system.

Customer
Order UI



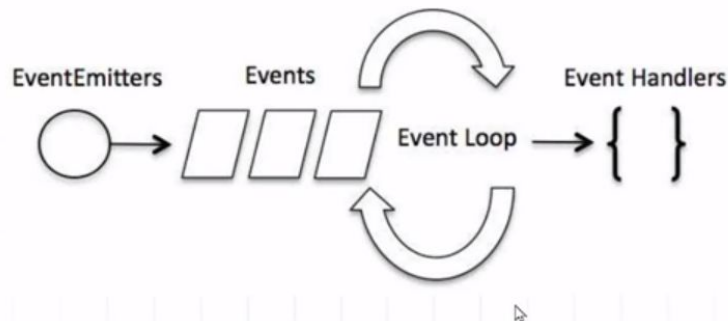
0:00 / 0:30

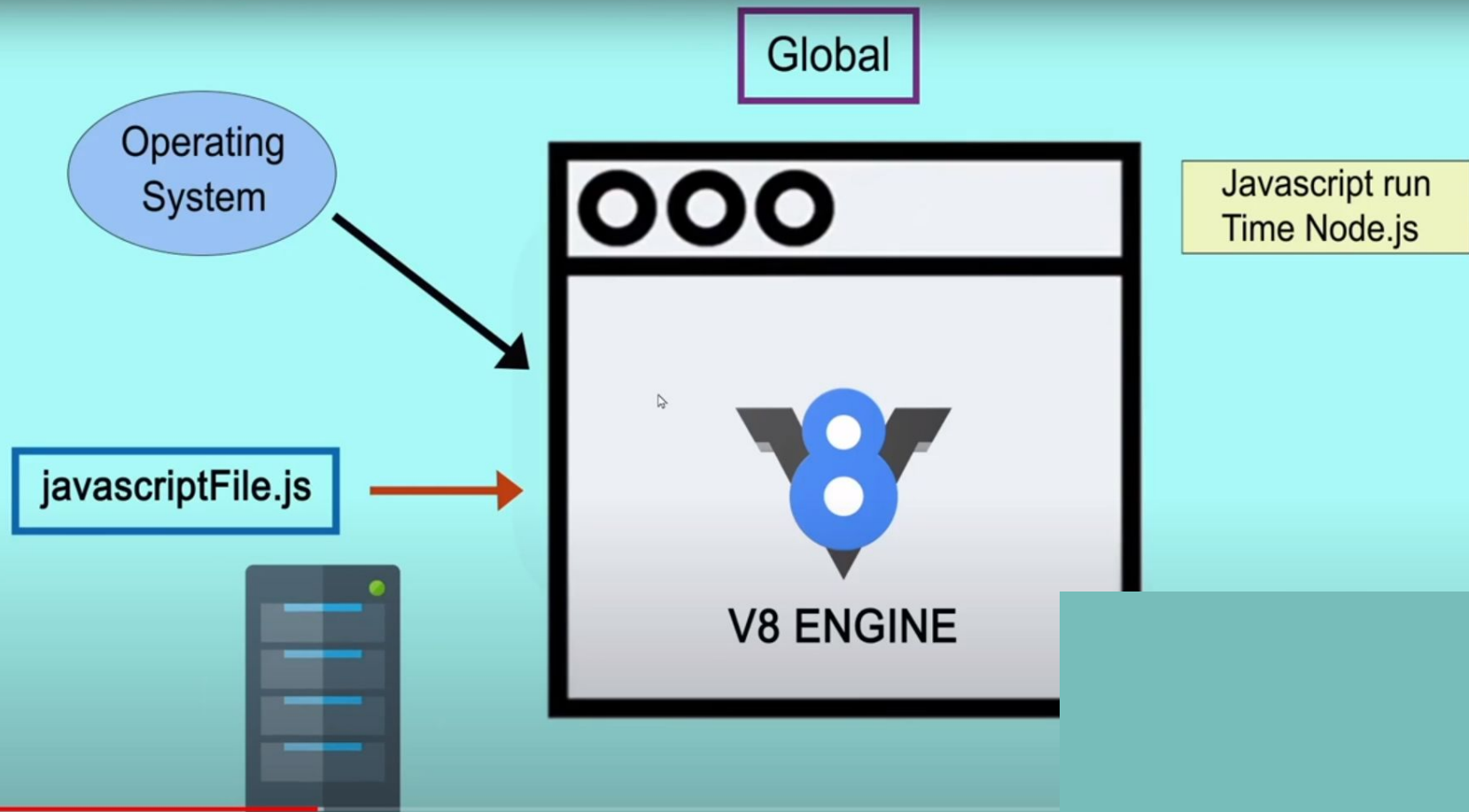


Event-driven in Node JS

The basic components:

- A **callback function** (called an event handler) is called when an event is triggered.
- An **event loop** that listens for event triggers and calls the corresponding event handler for that event.
- A function that listens for the triggering of an event is said to be an '**Observer**'. It gets triggered when an event occurs.
- Most of the in-built modules of Node.js inherit from the **EventEmitter** class. **The EventEmitter is a Node module that allows objects to communicate with one another. The core of Node's asynchronous event-driven architecture is EventEmitter.**





Server with Threads and Blocking Behaviour

4

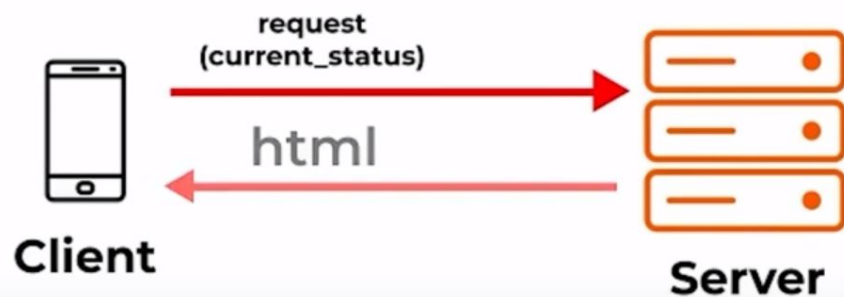


Client

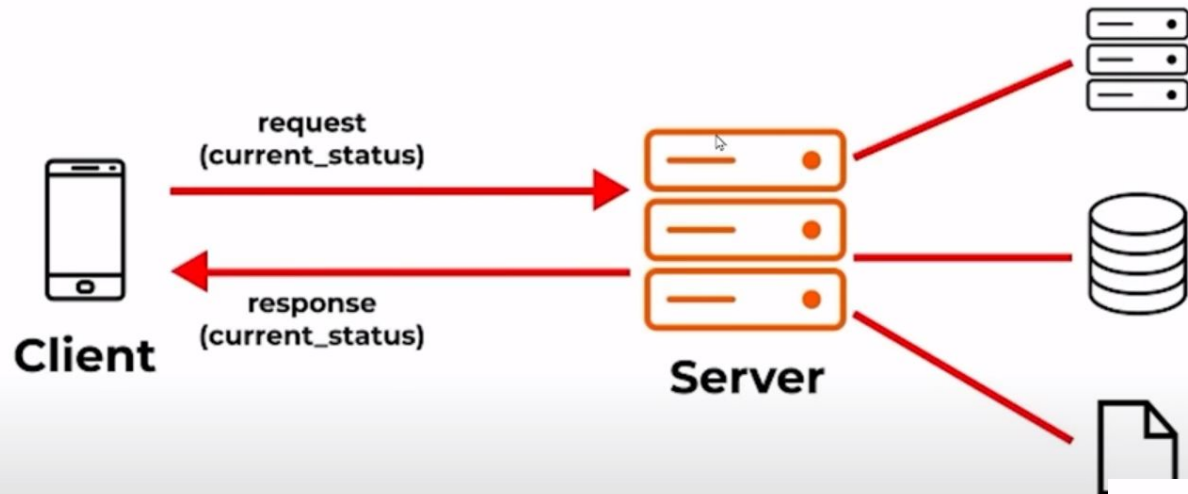


Server

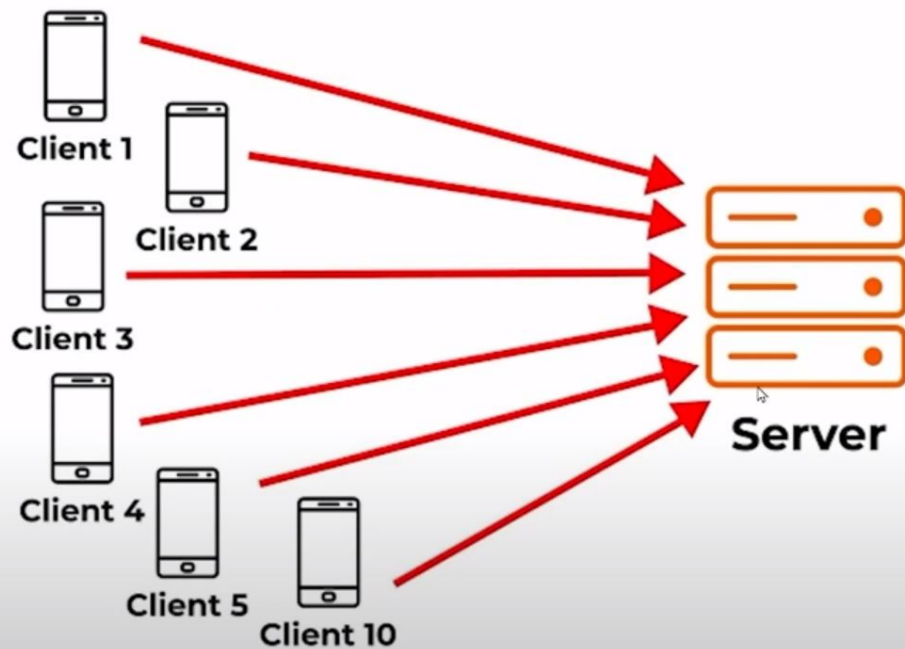
Server with Threads and Blocking Behaviour



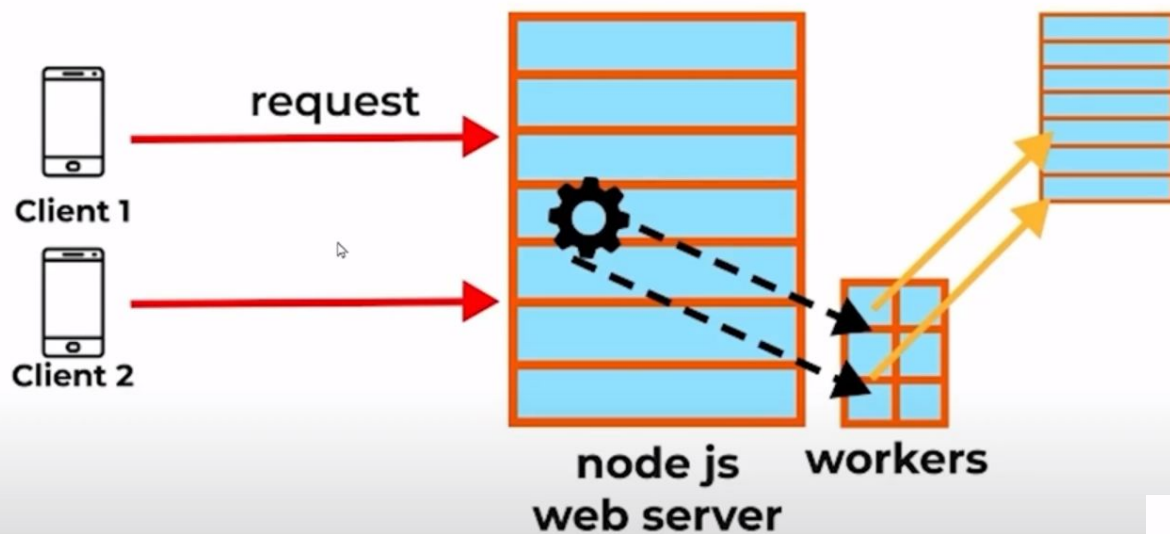
Server with Threads and Blocking Behaviour



Server with Threads and Blocking Behaviour



Server with Threads and Blocking Behaviour



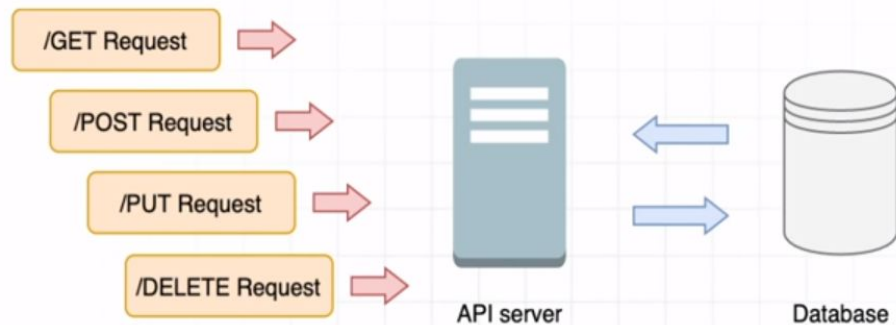
When to use Node JS

API Application - A great choice for constructing an API application with both relational and non-relational databases.

It's all because:

- Node.js runs on a single thread which makes it easier to handle up to 10,000 concurrent requests
- All blocking I/O tasks (i.e. database access) are always being processed asynchronously by internal threads without interrupting the main thread

This makes **Node.js** good to handle the requests, make database operations, and expose JSON objects for clients.



When to use Node JS

Real time Applications

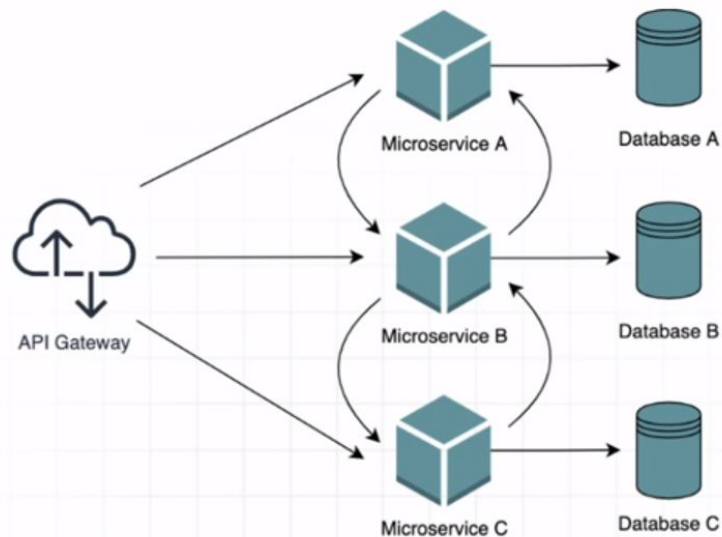
Good at building real-time applications like

- messaging,
- notifications delivery,
- live streaming and
- collaboration tools.

When to use Node JS

Microservices

Node can also be used to build microservices — an architectural approach based on building an application as a collection of small services. Each microservice has its own data model and manages its own data.

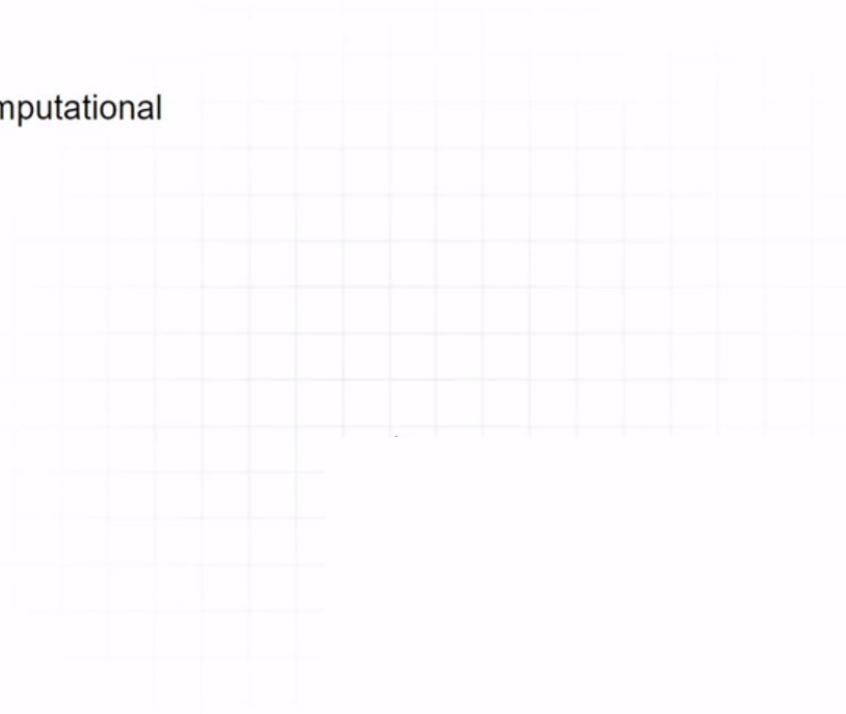


When to use Node JS (at a glance)



When not to use Node JS



- CPU intensive tasks/ heavy computational applications
 - Backend with relational databases
- 

Companies using Node JS



Real-time applications.

Node.js is able to maintain tough traffic with no loss in performance. This makes the environment a perfect choice for real-time messaging and chatting apps.

Collaborative drawing and editing tools (Google Docs, Figma, Sketch);

Live-chat apps (LiveChat, Smartsupp)



Online gaming apps.

As Node.js delivers a real-time and dynamic experience, online games are one of its leading use-cases (Ancient Beast, PaintWar, Voxel Shooting, Anagrammatix)



E-commerce transaction software.

Huge traffics of e-commerce platforms demand a proven technology to satisfy all users' requests (PayPal).



Video-Conferencing apps.

Node.js will work here with specific hardware or VoIP (Zoom, Google Hangouts, Skype).



Thank you!