zeolearn™

Module 1
Introduction to node.js

zeolearn™

Outline

- An overview of Node.js
- Installing Node.js
- Preparing development environment
- Quick peek into Node’s event loop
- Writing asynchronous code with callbacks
- Debug first node app

zeolearn™

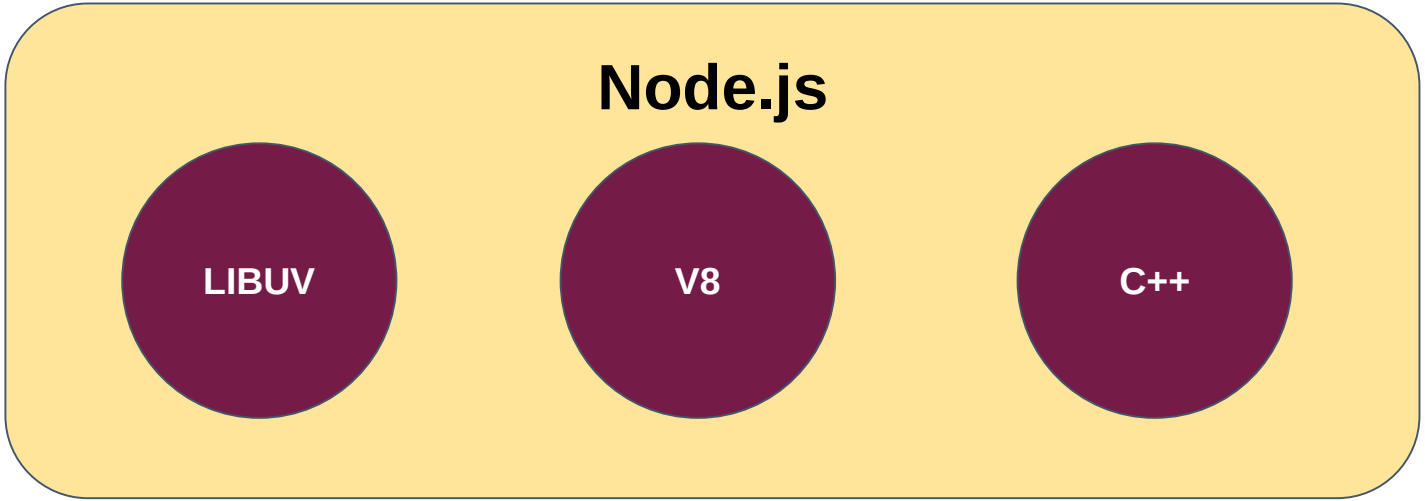
What is nodejs - Official website says

- Node.js® is a **JavaScript runtime** built on Chrome's V8 JavaScript engine.
- Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.

Where is node used

- Server side development
- Desktop and IoT development

Core components



Setup Node

- Download binary of targeted platform and install
- https://nodejs.org/en/download/
- Test your installation
  - Open terminal run node -v
  - Open terminal run npm -v

zeolearn™

Preparing development environment

- Any editor of choice (ST3, NP++, WS, VSCode, Atom)
- Command line

Welcome to node

```
node -v
```

```
npm -v
```

Node REPL

- **R**ead **E**valuate **P**rint **L**oop
- Command line tool to quickly evaluate / experiment your work

- **ctrl + c** – terminate the current command.
- **ctrl + c twice** – terminate the Node REPL.
- **ctrl + d** – terminate the Node REPL.
- **Up/Down Keys** – see command history and modify previous commands.
- **tab Keys** – list of current commands.
- **.help** – list of all commands.
- **.break** – exit from multiline expression.
- **.clear** – exit from multiline expression.
- **.save *filename*** – save the current Node REPL session to a file.
- **.load *filename*** – load file content in current Node REPL session.

Helloworld Node

Filename : helloworld.js

```
let greeting = 'Helloworld Node';
console.log(greeting);
```

run code : **node helloworld.js**

Execution model

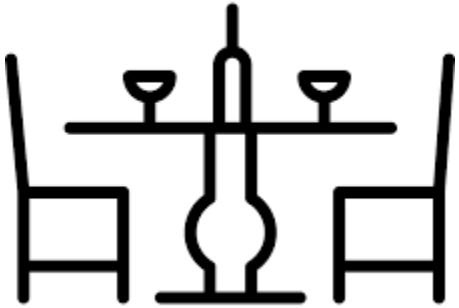
- Non-blocking
  - **Non-blocking I/O, asynchronous I/O, or "Non-sequential I/O"** is a form of input/output processing that permits other processing to continue before the transmission has finished.
- Event Driven
  - A programming paradigm in which the flow of the program is determined by **events** such as user actions or **messages** from other **programs/threads**

Why so much fuss about sync vs async

- Javascript is single threaded, so nodejs runtime. Which means JS can do ONE AND ONLY THING AT A TIME.

zeolearn™

Restaurant Example



3 tables
(users)

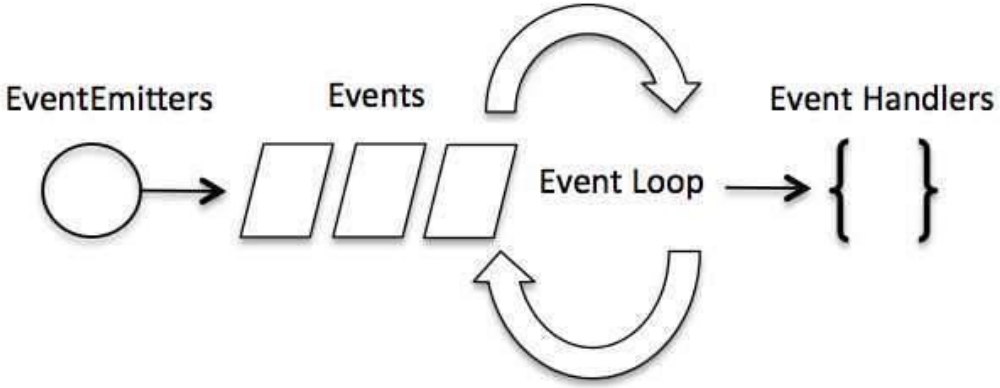
1 waiter
(node)

2 chefs
(db, fs)

Event loop - The HEART & SOUL

- The event loop got its name because of how it's usually implemented
- Responsible for scheduling asynchronous operations

```
while (queue.waitForMessage())
  queue.processNextMessage()
}
```



Debug node application

```
node --inspect-brk helloworld.js
chrome://inspect
```

Summary

- What is nodejs
- Setting up development environment
- Non blocking I/O
- Eventloop
- Debugging node apps

zeolearn™

Check your knowledge

- What is Server side runtime for javascript?
- Which Javascript vm is used by node?
- Name 2 important design decisions of node runtime
- Which command is used to run code in node?
- Expand REPL
- Descramble - **vneotpeol**