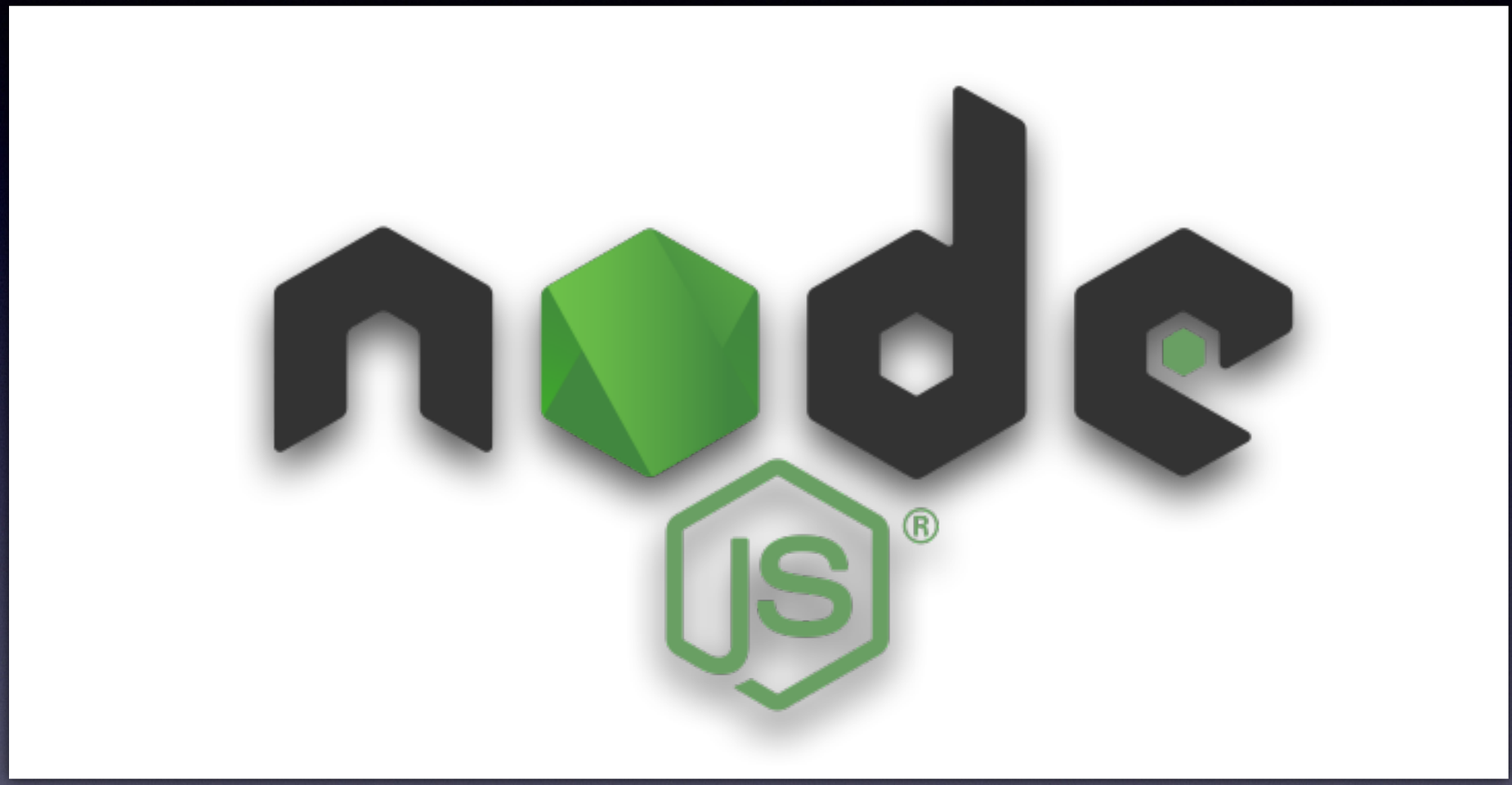
Node.JS

Arun Vijayarengan

Node.JS Intro



A JS Runtime runtime built on Chrome V8

JS Prerequisites

- In JavaScript
    - Variable Hoisting
    - Events and Callbacks
    - Object and JSON
    - let, const, Arrow Functions, Template Literal

Essential JS

• Variable Hoisting	• http://jsbin.com/fuyahi/2/edit?js,console
• Events	• http://jsbin.com/niqifo/24/edit?html,js,console,output
• Callbacks	• http://jsbin.com/tamaqak/1/edit?js,console
• Objects	• http://jsbin.com/xubego/23/edit?js,console,output
• JSON	• https://jsonplaceholder.typicode.com/users
• let	• https://jsonplaceholder.typicode.com/users
• const	• http://jsbin.com/tazizuf/edit?js,console
• Template Literal	• http://jsbin.com/zevaqay/3/edit?js,console
• Arrow Functions	• http://jsbin.com/bawibax/2/edit?js,console

V8 Engine

- Google's open source high-performance JavaScript engine
- written in C++; used in Google Chrome and Node.JS
- V8 compiles JavaScript directly to native machine code before executing it

JS in Server Side

What JS needs to become Server Side Script?

- JS needs to Accept Requests and Send Responses
- JS needs to communicate over Internet
- JS should deal with Files and File System
- JS should deal with Databases

Synchronous

- JS in Synchronous
- Executes only one process at a time
- V8 is also Synchronous

Asynchronous

- Executing more than one process simultaneously

- Node.JS is Asynchronous

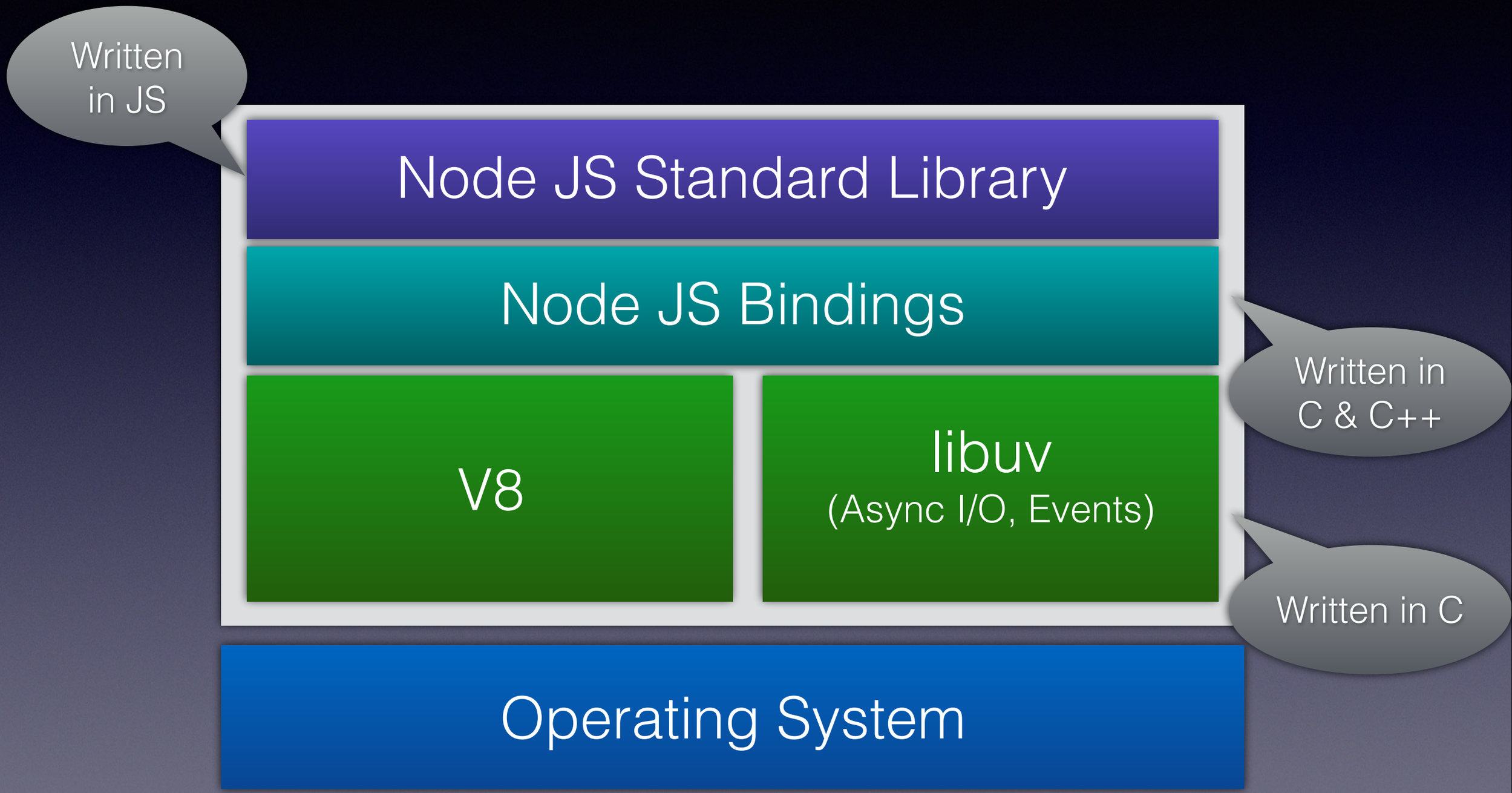
First Node Program

Running in NodeJS Terminal

```
> node

    > console.log("Success");
```

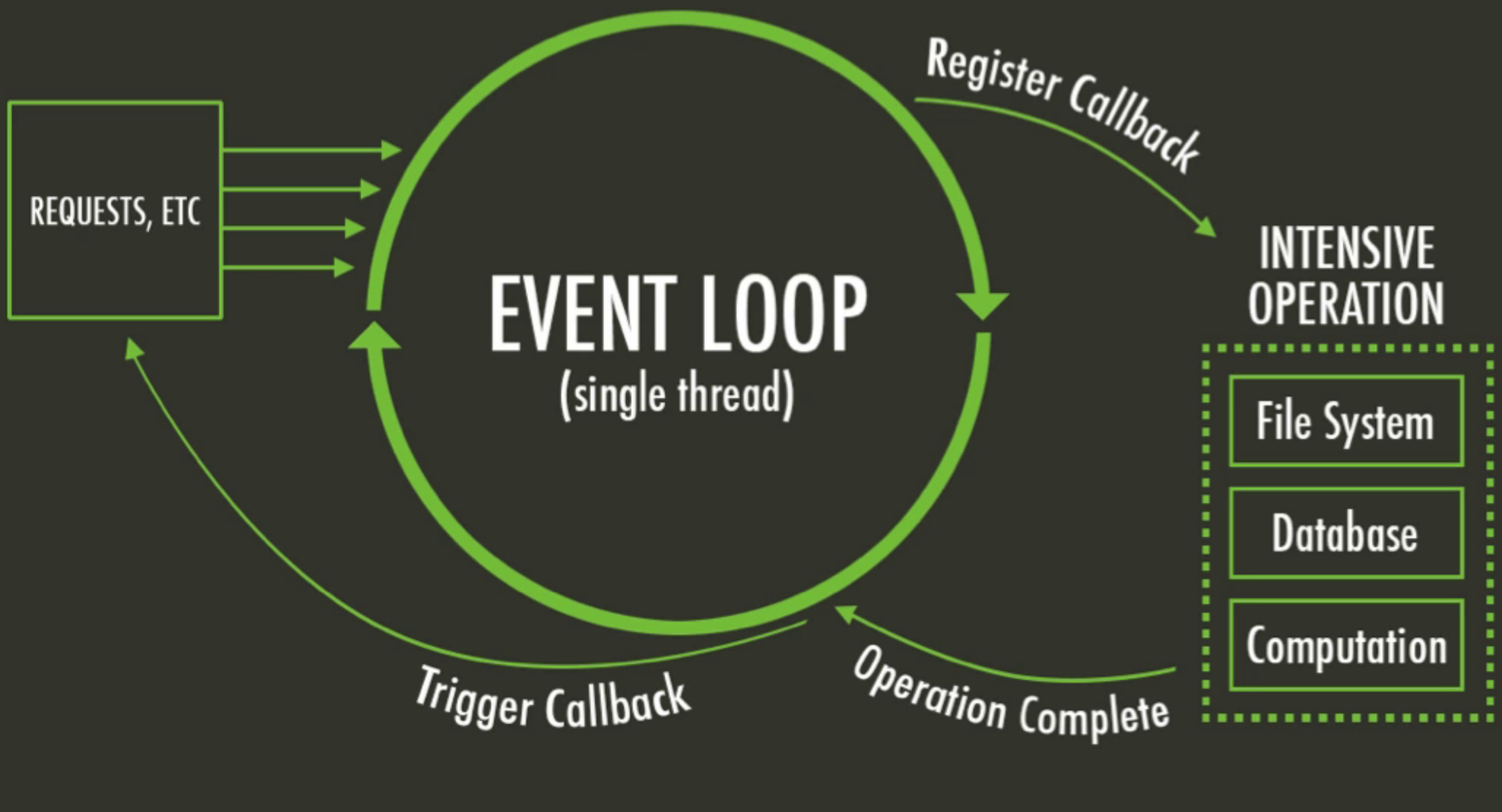
Node.JS Architecture



Events and Event Emitter

```
var fs = require('fs');
var file = fs.createReadStream('./test.txt');
file.on('error', function(err) {
  console.log('Error '+err);
  throw err;
});
file.on('data', function(data) {
  console.log('Data '+data);
});
file.on('end', function(){
  console.log('Finished reading all of the data');
});
```

Non-blocking I/O



Blocking or Sync I/O

Demo

Non-blocking or Async I/O

Demo

Event Loop & Event Emitter

Demo

Introduction to NPM

- Node Package Manager
- http://npmjs.com - A Central repository of Node.JS Packages
- Install package(s) using

  'npm install package-name'

Creating a new Node Project

```
> npm init
```

package.json

walkthrough

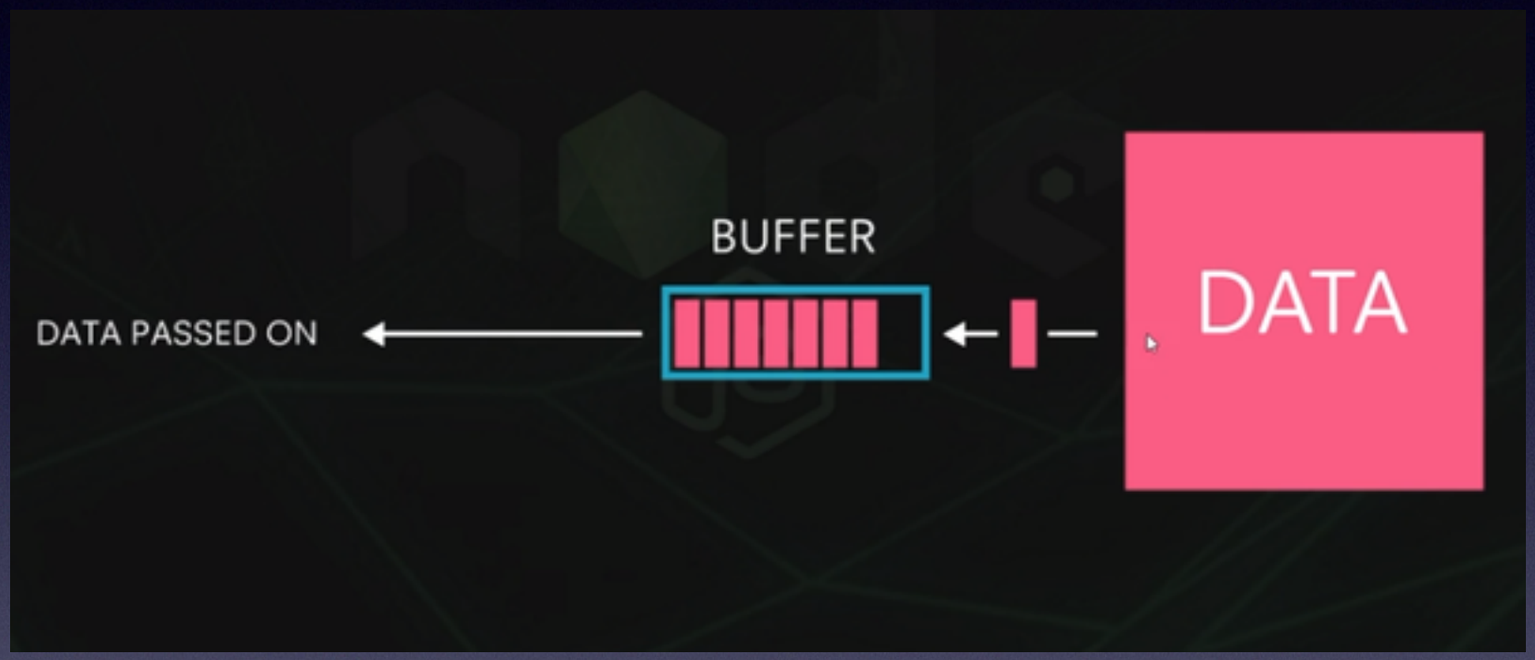
File System

- var fs = require('fs');
    - fs.path()
    - fs.mkdir()
    - fs.rmdir()
    - fs.rename()
    - fs.open()
    - fs.read()
    - fs.write() and many more are there

Buffers

- Buffer is designed to handle raw binary data.
- Buffer is a container outside V8 (raw memory allocated) for storing raw bytes
- A byte = eight bits, and a bit is just a 0 or a 1. So a byte might look like 10101010
- The Buffer class is a global class in Node.js

Buffer - Illustration



Example: Buffering Videos

Buffer Example

Demo

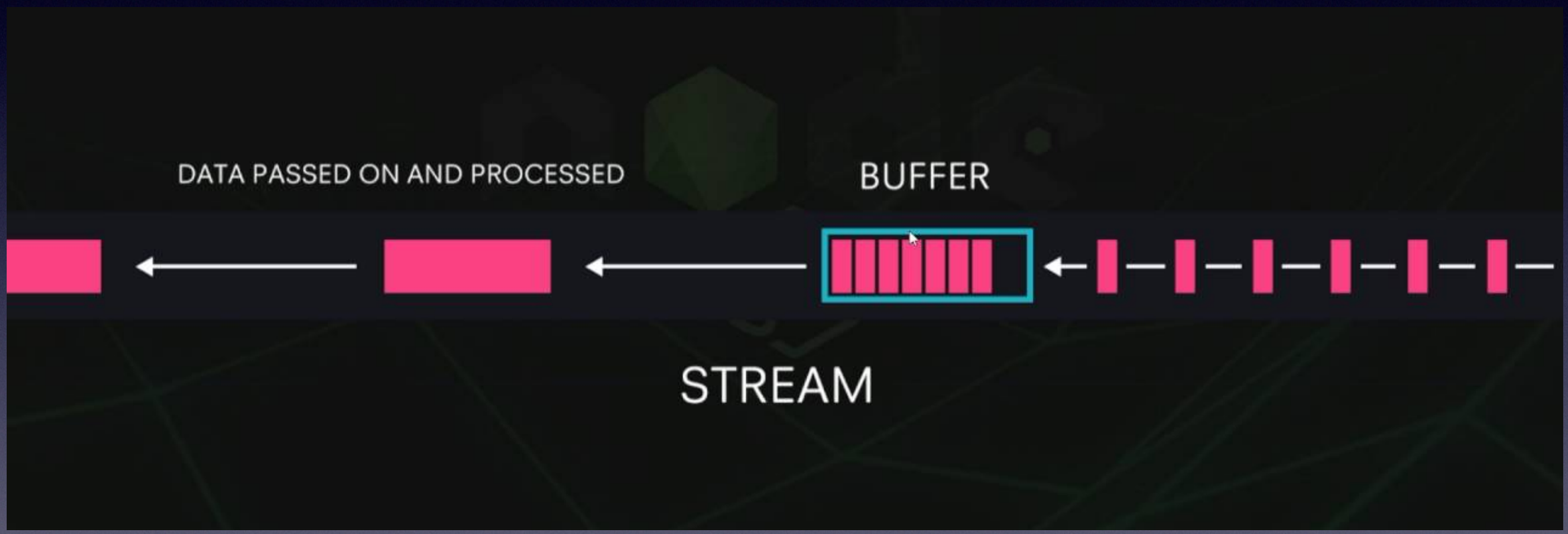
Streams

- A stream is an interface for working with streaming data in Node.js.

- The stream module provides a base API that makes it easy to build objects that implement the stream interface.

- Streams are objects that let you read data from a source or write data to a destination in **continuous** fashion.

Streams - Illustration



Types of Streams API's

Readable	Writable	Duplex	Transform
Streams used for read operation	Streams used for write operation	Streams used for both.	Output is in some way related to the input

Common Stream Events

- data -- when the data is available to read
- end -- when there is nothing more to read
- error -- when the reading/writing resulted in error
- finish -- when the operation is over

Socket

- A socket is one endpoint of a two-way communication link between two programs running on the network.

- A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

- An endpoint is a combination of an IP address and a port number.

Socket.IO

- Socket.IO is a JavaScript library for realtime web applications.
- It enables realtime, bi-directional communication between web clients and servers.
- It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js.
- Learn more: https://en.wikipedia.org/wiki/Socket.IO

Building Socket.IO App

Demo

Clustering

- Node.JS uses only only core in a machine. It's designed in that way.
- Even in Dual Core, Quad Core, 8 Core processors Node.JS just uses one Core.
- Clustering helps us run many copies of the program in many cores. (ex: running 4 programs in quad core machines, i.e: the machine can handle 4 times the traffic)
- Clustering is a great way to add stability to your application and drastically increase it's load capacity.

Processors and Cores

Multi-core processors

- **Dual-core processor** has two cores
  (AMD Phenom II X2, Intel Core Duo)
- **Quad-core processor** contains four cores
  (AMD Phenom II X4, Intel's quad-core processors, see i3, i5, and i7 at Intel Core).
- **Hexa-core processor** contains six cores
  (AMD Phenom II X6, Intel Core i7 Extreme Edition 980X)
- ***Octa-core processor*** contains eight cores
  (Intel Xeon E7-2820, AMD FX-8150).

How to Cluster Express App?

- Learn from here: https://stackoverflow.com/questions/34868258/how-to-use-node-cluster-module-within-express-generator-app-skeleton

- And here: http://rowanmanning.com/posts/node-cluster-and-express/

Spawn

- Spawn in computing refers to a function that loads and executes a new child process. The current process may wait for the child to terminate or may continue to execute concurrent computing. Creating a new subprocess requires enough memory in which both the child process and the current program can execute.

Using PM2 Module for Clustering

- Refer: http://pm2.keymetrics.io/
- Also: http://pm2.keymetrics.io/docs/usage/cluster-mode/