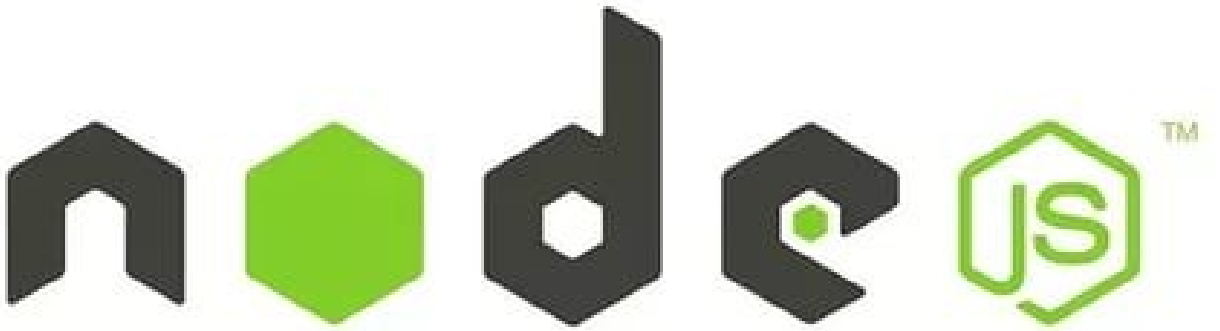

node
JS
TM

# Node.js

- Применение Node.js
- Архитектура Node.js
- Запуск HTTP сервера

# 1 | Применение

- Одностраничные приложения (AJAX-приложение вроде Gmail). Обработка тысячи запросов с низким временем отклика
- Системы мягкого реального времени. Такие приложения как блоги, чаты, системы бронирования online, игры и т.д.
- JSON API  - Node.js блестяще справляется с построением легковесных REST / JSON интерфейсов. Отличный вариант для написания обертки вокруг базы данных или веб-сервиса, которая общается с клиентом в формате JSON.
- Консольное приложение

- **Применение лишено смысла в приложениях с длительными сложными вычислениями и небольшим количеством операций ввода-вывода. Так что если вы собираетесь писать перекодировщик видео, искусственный интеллект, солвер  пожалуйста, не используйте Node.js**

# **2 |** Архитектура

Node.js — серверная платформа в основе которой лежит реализация языка программирования JavaScript, основанная на движке V8. Предназначена для создания  сетевых приложений, таких как веб-сервер.

Node.js использует событийный цикл (eventdriven) ,не блокирующую модель ввода/вывода(I/O), которая создает эффективное взаимодействие с распределенными устройствами.

Node.js — работает в режиме однопоточного сервера

# 2 | Blocking Code Example

Создадим текстовый файл  index.htm со следующим содержимым:
<html> <boby><h1>Hello 1</h1> </body>

**main.js:**

```
var fs = require("fs");
var data = fs.readFileSync('index.html');
console.log(data.toString());
console.log("Program Ended");
```

**Console:**

```
$ node main.js
```

**Output:**

```
<html> <boby><h1>Hello 1</h1></body>
Program Ended
```

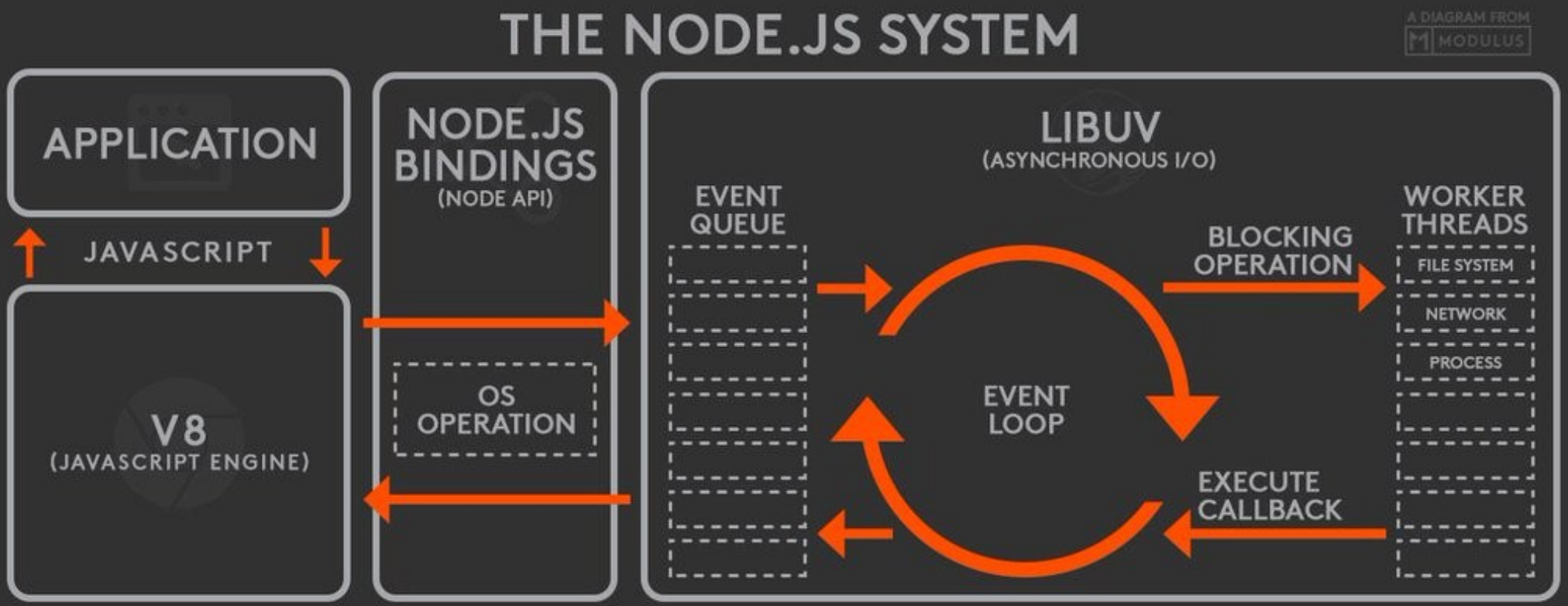
# 2 | Non blocking Code Example

```
var fs = require("fs");
 fs.readFile('input.txt', function (err, data) {
      if (err) return console.error(err);
          console.log(data.toString());
   });
  console.log("Program Ended");
```

```
$ node main.js
Output:
Program Ended
<html> <boby><h1>Hello 1</h1> </body>
```

## **2 |** Архитектура

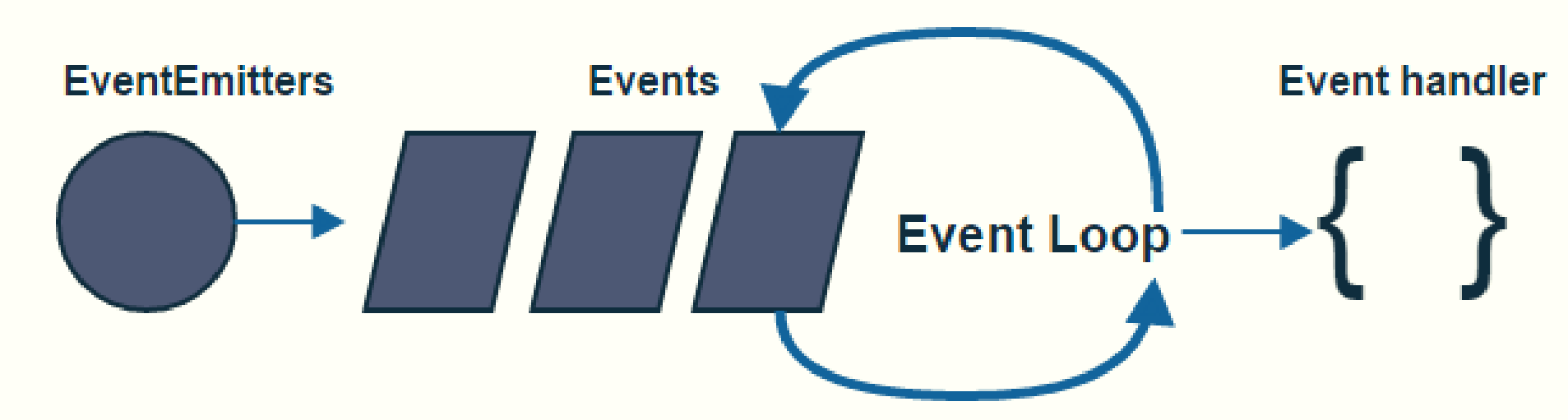
В основе Node.js лежит библиотека libev, реализующая цикл событий (event loop). Libev — это написанная на C библиотека событийно-ориентированной обработки данных, предназначенная для упрощения асинхронного неблокирующего ввода/вывода.

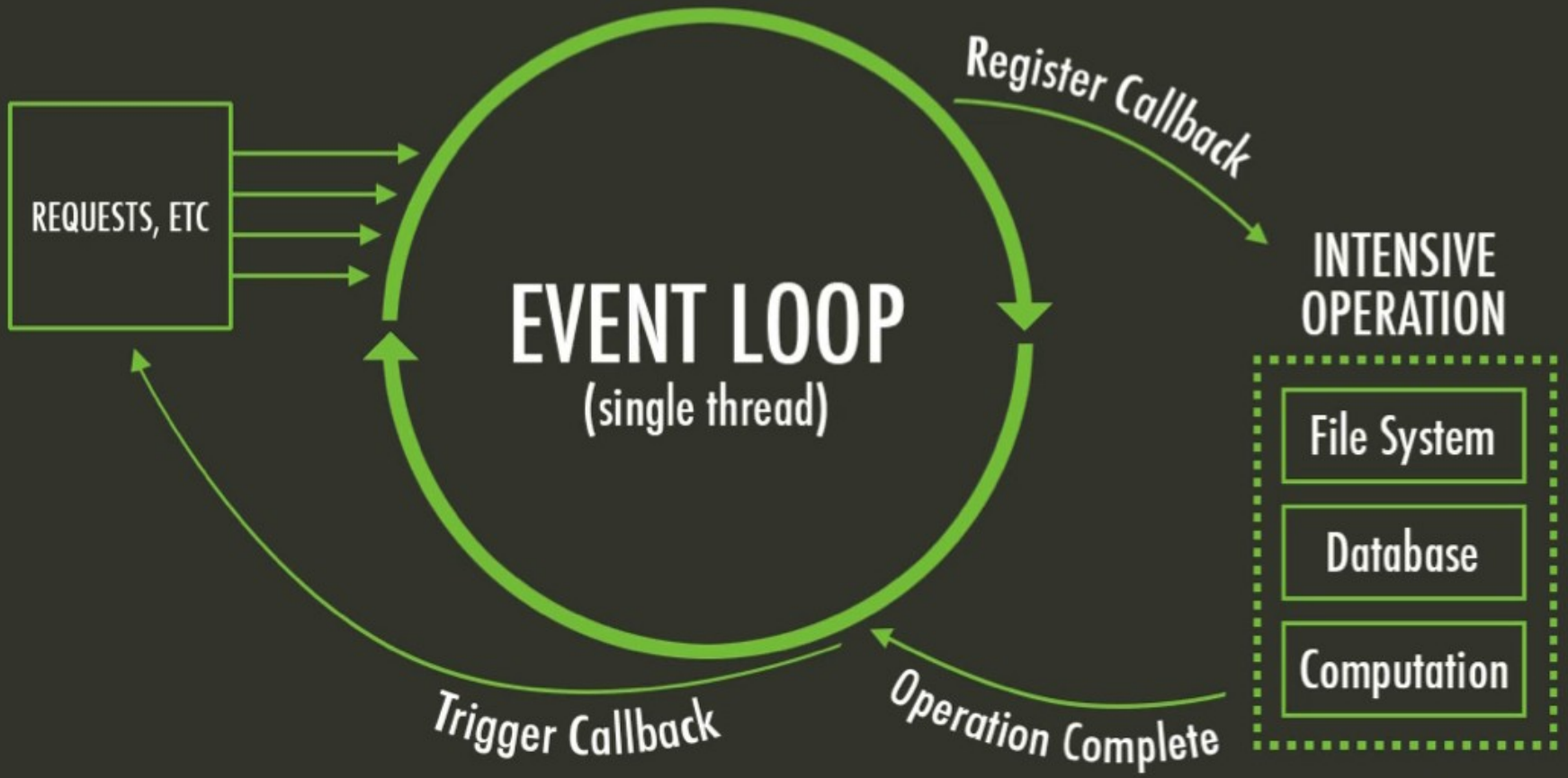

THE NODE.JS SYSTEM
A DIAGRAM FROM MODULUS
APPLICATION
JAVASCRIPT
V8
(JAVASCRIPT ENGINE)
NODE.JS BINDINGS
(NODE API)
OS OPERATION
LIBUV
(ASYNCHRONOUS I/O)
EVENT QUEUE
EVENT LOOP
BLOCKING OPERATION
EXECUTE CALLBACK
WORKER THREADS
FILE SYSTEM
NETWORK
PROCESS

# 2 | Библиотека libev

Библиотека libev реализует цикл событий (event loop). Это написанная на C библиотека событийно-ориентированной обработки данных, предназначенная для упрощения асинхронного неблокирующего ввода/вывода

Event Loop — прослушивает события и вызывает функцию обратного вызова(callback function) для данного события когда оно произошло.




REQUESTS, ETC
EVENT LOOP
(single thread)
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Trigger Callback
Operation Complete

# Что такое callback ?

Мы хотим чтобы функция выполнилась через 1 секунду

```
function doSomething() {
    alert("прошла секунда")
}
setTimeout(doSomething, 1000);
```

А как сделать чтобы эта функция вызывалась по наступлению определенного события ? К примеру событие click на кнопке ?

# Что такое callback ?

Мы создаем сервер и передаем функцию в созданный им метод. Всякий раз, когда наш сервер получает запрос, переданная нами функция будет вызываться.

```
var http = require("http");
function onRequest(request, response) {
   response.writeHead(200, {"Content-Type": "text/plain"});
   response.write("Hello World");
   response.end();
}
http.createServer(onRequest).listen(8888);
```

Этот принцип называется обратный вызов или callback. Мы передаем в некоторый метод функцию и этот метод исполняет её, когда происходит связанное с методом событие.

# EventEmmiter — основной объект, реализующий работу обработчиков событий в Node.js

```
// require the EventEmitter from the events module
const EventEmitter = require('events').EventEmitter

// create an instance of the EventEmitter object
const eventEmitter = new EventEmitter()

// register a listener for the 'randomString' event
eventEmitter.on('randomString', function (randomStr) {
  console.log('Received the string: ' + randomStr)
})

// trigger an event called 'randomString' and send
// a randomly selected string to the listeners
eventEmitter.emit('randomString', randomString())

// simple function to randomly select a string from an array
function randomString () {
  const stringsArr = ['NodeJs', 'coligo.io', 'JavaScript', 'EventEmitters']
  return stringsArr[Math.floor(Math.random() * stringsArr.length)]
}
```

# Как выйти из цикла

process.exit()

process.nextTick()

process.kill()

# Полезный ссылки

http://nodebeginner.ru/ - Node.js для начинающих

https://nodejs.org - официальный сайт