FULL STACK WEB DEVELOPMENT – II

ASSIGNMENT 1

KASHISH MITTAL
22105006

NODE JS

- What is node js ?
- Node js architecture
- How node js is single threaded ?
- How node js is asynchronous ?
- How node js is event driven ? *(ALL AT SAME TIME)*

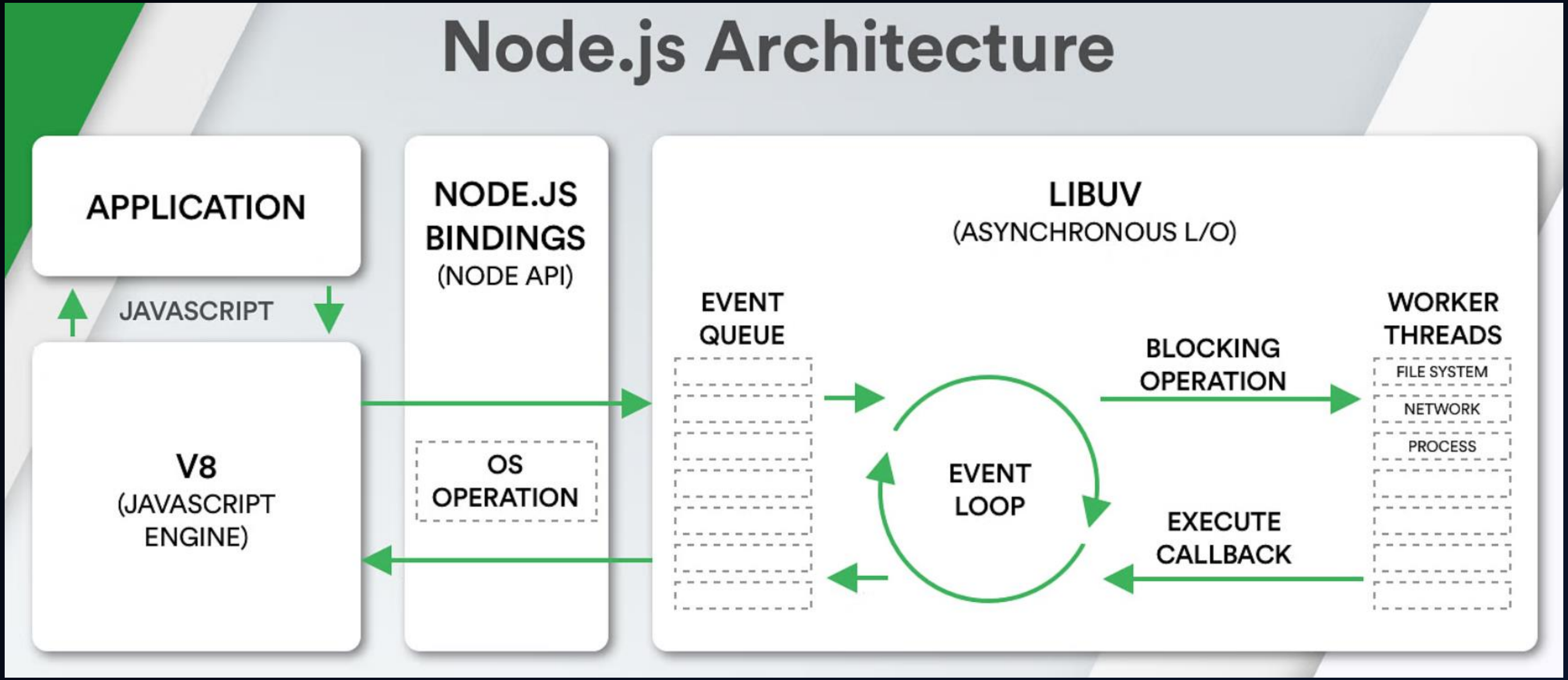
WHAT IS NODE JS

Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside of a web browser. It uses a non-blocking, event-driven I/O model, making it efficient for building highly scalable network applications.

Key features of Node.js:

- **Asynchronous programming**
- **Single-threaded**
- **Non-blocking I/O**
- **Cross-platform**
- **Large ecosystem**

NODE JS ARCHITECTURE



NODE JS IS **SINGLE THREADED,** ASYNCHRONOUS ,AND EVENT DRIVEN AT SAME TIME

Node.js achieves this unique combination of single-threaded, asynchronous, and event-driven programming through its efficient use of non-blocking I/O operations and an event loop.

Single-Threaded:

- Node.js uses a single thread to handle all incoming requests. This simplifies development and reduces overhead compared to multi-threaded models.

NODE JS IS **ASYNCHRONOUS** ,SINGLE THREADED AND EVENT DRIVEN AT SAME TIME

Asynchronous:

- Node.js uses non-blocking I/O operations, meaning it doesn't wait for an operation to complete before moving on to the next task. This allows it to handle many concurrent requests without blocking the main thread.
- When an asynchronous operation (like reading a file or making a network request) is initiated, Node.js provides a callback function that will be executed when the operation completes.

NODE JS IS NODE JS IS **EVENT DRIVEN,** ASYNCHRONOUS AND SINGLE THREADED AT SAME TIME

Event-Driven:

- Node.js uses an event loop to manage asynchronous operations. This loop continuously checks for new events and executes the corresponding callbacks when they occur.
- When an asynchronous operation completes, its callback function is added to an event queue.
- The event loop periodically checks the queue for new callbacks and executes them.

HOW IT ALL WORKS TOGETHER:

1. **Incoming Request:** A client sends a request to a Node.js server.
2. **Event Loop:** The event loop receives the request and adds it to the event queue.
3. **Asynchronous Operation:** If the request involves an asynchronous operation (like reading a file from the disk), Node.js delegates the operation to the operating system and continues processing other events in the event loop.
4. **Callback Queue:** When the asynchronous operation completes, its callback function is added to the callback queue.
5. **Event Loop Processing:** The event loop eventually processes the callback queue and executes the callback function. The callback function can then handle the result of the asynchronous operation and send a response to the client.