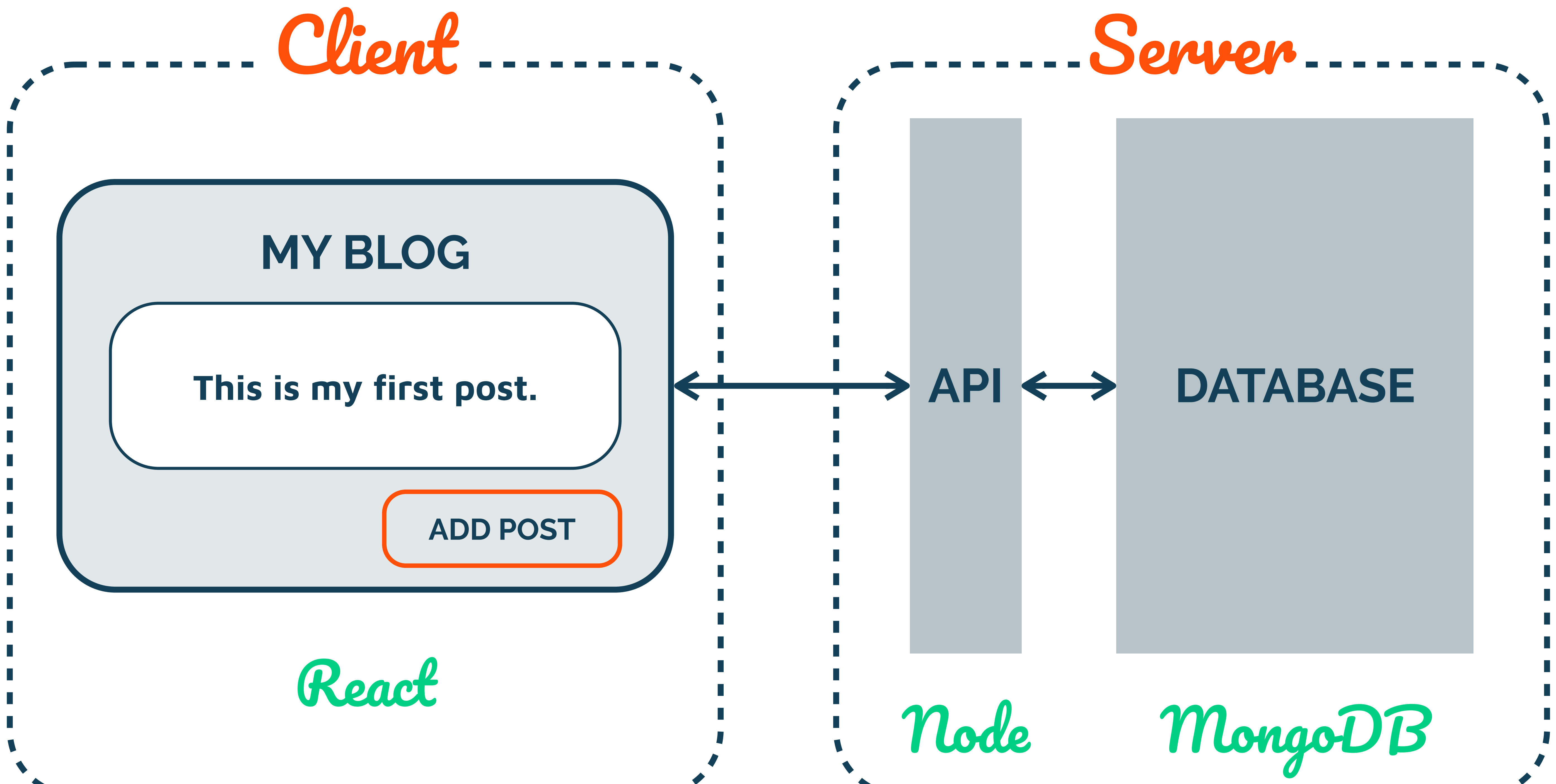
BACKEND


Client
MY BLOG
This is my first post.
ADD POST
React
Server
API
DATABASE
Node
MongoDB

NODE.JS

NODE.JS

“Node.js is a platform built on ***Chrome’s JavaScript runtime*** *for easily building fast, scalable network applications. Node.js uses an* ***event-driven****,* ***non-blocking I/O model*** *that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices”*

https://nodejs.org/en/about/

NON-BLOCKING

Enables handling **many concurrent connections**

I/O operations are generally slow

Upon each connection, the **handler** is fired

While one process is waiting for I/ O, let another process make use of CPU

https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/

TRADITIONAL SERVERS

Servers serve several clients at the same time: how?

Multi-process or **multi-threaded**

Each connection results in the creation of a **dedicated child process/thread**

Parent process/main thread remains available, listening for new connections

http://www.baloo.io/blog/2013/11/30/node-event-driven-programming/

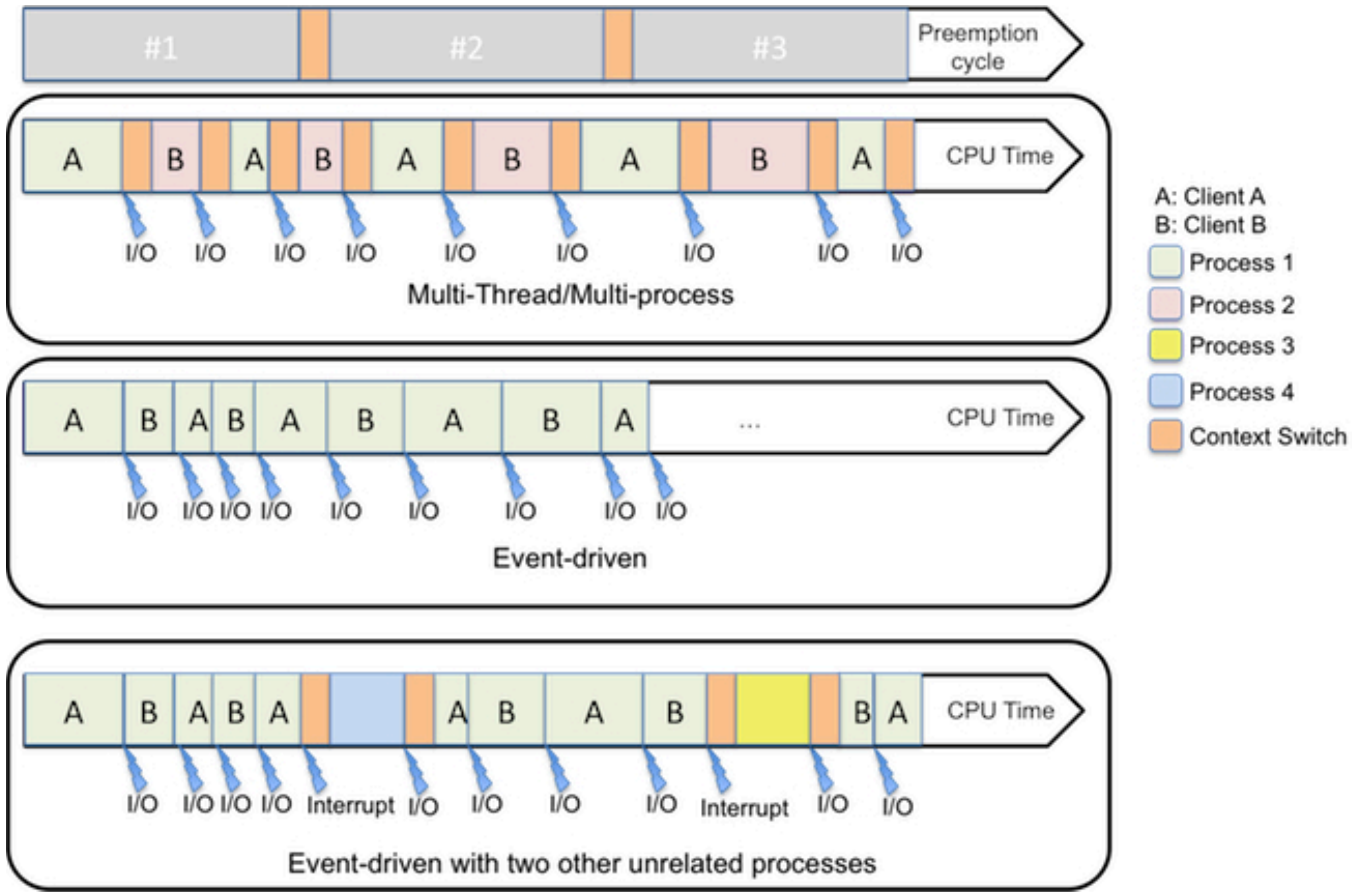
EVENT-DRIVEN CONCURRENCY

Everything runs in **one process**, one thread

An event is emitted and the appropriate **callback** for that event is invoked

Once an event is treated, the process is ready to treat another event

http://www.baloo.io/blog/2013/11/30/node-event-driven-programming/



http://www.baloo.io/blog/2013/11/30/node-event-driven-programming/

THE EVENT LOOP

All the events are processed by **event-loop queue.**

Event-loop fetches next event to process and dispatches the corresponding handler

Anyone blocking the event-loop will prevent the other events from being processed

Single-threaded: **DO NOT BLOCK THE EVENT LOOP**

Node API is **non-blocking** (with the exception of some file system operations which come in two flavors: asynchronous and synchronous)

https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/

JAVASCRIPT AND I/O

JavaScript was designed for being used inside a browser; missing basic I/O libraries (such as file operations)

Node: Javascript + I/O API
could make I/O natively non-blocking in Node

https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/

BASIC EXAMPLE

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

https://nodejs.org/en/about/

EXPRESS

EXPRESS

Minimal and flexible Node.js **web application framework** that provides a robust set of features for web and mobile applications.

HTTP utility methods and middleware for **building APIs**

"a thin layer of fundamental web application features, without obscuring Node.js features that you know and love."

https://expressjs.com/

INSTALLING EXPRESS

1 Create a new folder for your project

```
$ mkdir myapp
$ cd myapp
```

2 Initialize your Node project

```
$ npm init
...
```

3 Install Express using npm

```
$ npm install express
```

https://expressjs.com/en/starter/installing.html

HELLO WORLD

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => {
  console.log(`Example app listening on port ${port}!`)
})
```

https://expressjs.com/en/starter/hello-world.html

ROUTING

Structure

```
app.METHOD(PATH, HANDLER)
```

```
app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.post('/', function (req, res) {
  res.send('Got a POST request')
})

app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user')
})

app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user')
})
```

https://expressjs.com/en/starter/basic-routing.html

STATIC FILES

mount path

```
app.use('/static', express.static(path.join(__dirname, 'public')))
```

directory

https://expressjs.com/en/starter/static-files.html

MONGOOSE

MONGOOSE

```
$ npm install mongoose
```

Module for interacting with MongoDB

Provides a "better" interface than the official MongoDB driver

Schema-based solution to model app data

Built-in type casting, validation, query building, business logic hooks and more

https://mongoosejs.com/

SIMPLE APP WITH MONGOOSE

```
const readlineSync = require('readline-sync')
const mongoose = require('mongoose')

const Schema = mongoose.Schema

mongoose.connect('mongodb://localhost/test', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})

const db = mongoose.connection

const Llama = mongoose.model('Llama', new Schema({
  name: { type: String, required: true },
  age: { type: Number, required: true, min: [18, 'Adult llamas only!'] },
  created: { type: Date, default: Date.now }
}))
```

don't forget npm init & npm install

SIMPLE APP WITH MONGOOSE

```
async function loop() {
  const name = readlineSync.question('What is the llama\s name? ')
  const age = readlineSync.questionInt('How old is the llama? ')

  const llama = new Llama({ name, age })
  let result;
  try {
    result = await llama.save()
  } catch (err) {
    const errors = err.errors;
    Object.keys(errors).forEach(key => console.log(errors[key].message));
  }
  console.log(result)
  loop()
}

db.once('open', loop)
```

RESOURCES

https://nodejs.org/en/docs/guides/

https://nodejs.org/en/download/

https://expressjs.com/

https://www.postman.com/

https://mongoosejs.com/docs/guides.html

NEXT CLASS:
MIDTERM REVIEW

https://uiuc-web-programming.gitlab.io/fa21/