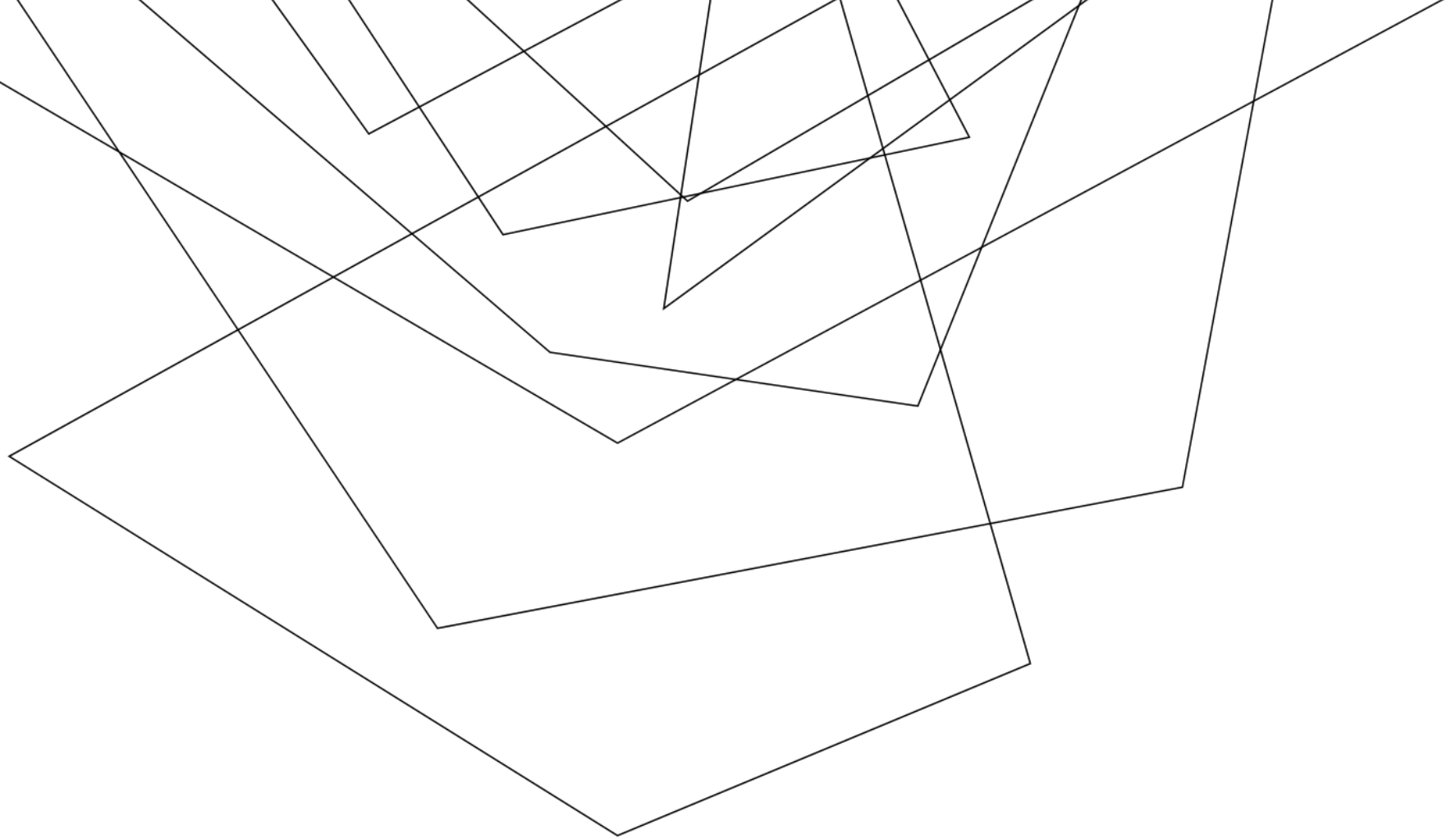
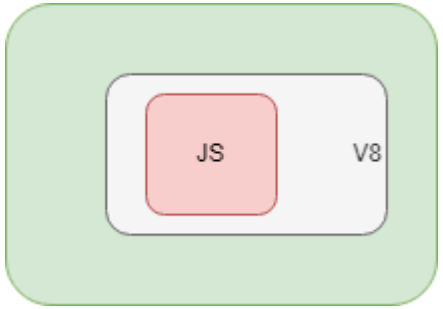


NODE JS

INTRODUCTION

- Is JavaScript runtime, built on Google's opensource JavaScript engine
- Is kind of container, inside which JavaScript code get executed, outside browser



- NodeJS got lot of libraries, which makes development easier
- Is JavaScript on server
- Can be extended as WebServer suitable for web deployment

INSTALLATION AND TOOLS

- Instructions available for installations on Windows or Linux flavor
- Visual Studio Code can be used for development
- NodeJS got lot of libraries, which makes development easier

Terminal demo on VSC and command line
\home\chandra\nodejs-learnings\first_prog.js

- Documentation available at -
https://nodejs.org/en/docs/

Installation on wsl ubuntu via node version manager (nvm)

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
source ~/.bashrc
nvm list-remote
nvm install v18.0.0
node -v
npm -v
npm install -g npm@latest
npm -v
```

MODULES

- Simple/Complex functionality organized in single/multiple JavaScript files which can be reused across applications
- This program demonstrate a module, which embed managing files specific functionality
  \home\chandra\nodejs-learnings\file_operations.js

A NOTE ABOUT UTF

- Machine understands 0 and 1 and Human understands characters
- ASCII is some standard established where each character is represented by combination of 0 and 1 –
  A = 01000001
  B = 01000010
- ASCII couldn't cover all characters
- Unicode Transformation Format - UTF is latest standard which covers everything

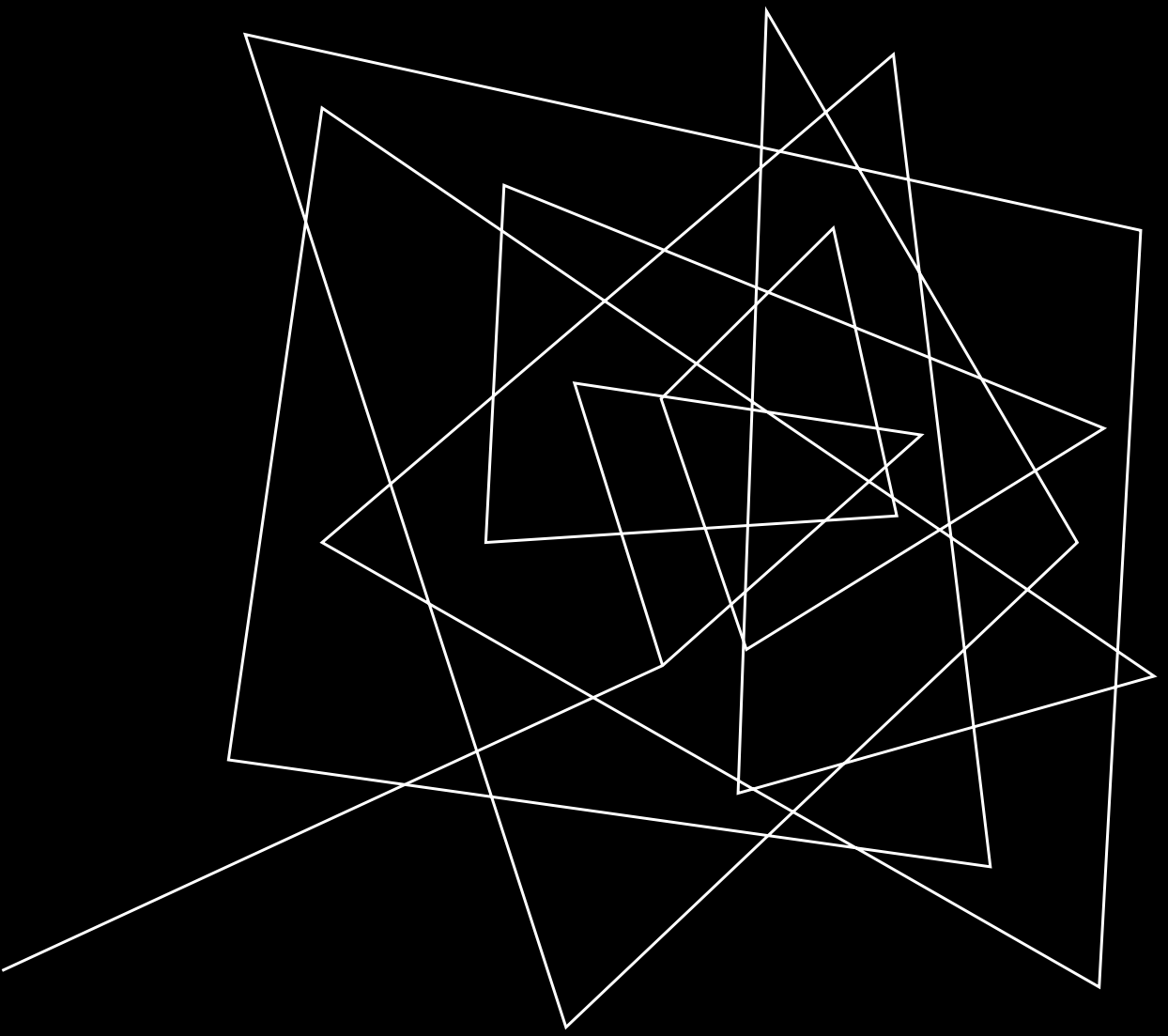
SYNCHRONOUS VS. ASYNCHRONOUS

- Synchronous calls, will process code lines one after another
- Each line blocks execution of next line and hence code is called as *blocking code*
- Can be issue when heavy task is running or task taking time
- Desired way is, the heavy task should run in background and other code be executing in parallel
- Such execution is *non-blocking* and is called as Asynchronous calls

HOW IT WORKS

- Callback is piece of code, a function, which gets called, when task is done
- NodeJS is Single threaded, event driven, non-blocking IO
- NodeJS Is light weighted and efficient
- NodeJS Is perfect for super fast application
- NodeJS Is not suitable to CPU intensive applications

\home\chandra\nodejs-learnings\file_operations_async.js



WEB SERVER

WEB SERVER

HTTP SERVER

- Works on *Request-Response* paradigm

\home\chandra\nodejs-learnings\http_server.js

- Will work on various details or req, res object in subsequent demos

ROUTING

- Different actions on different URLs

\home\chandra\nodejs-learnings\routing.js

- Will work on better routing solution in subsequent demos
- Meanwhile a better and efficient program

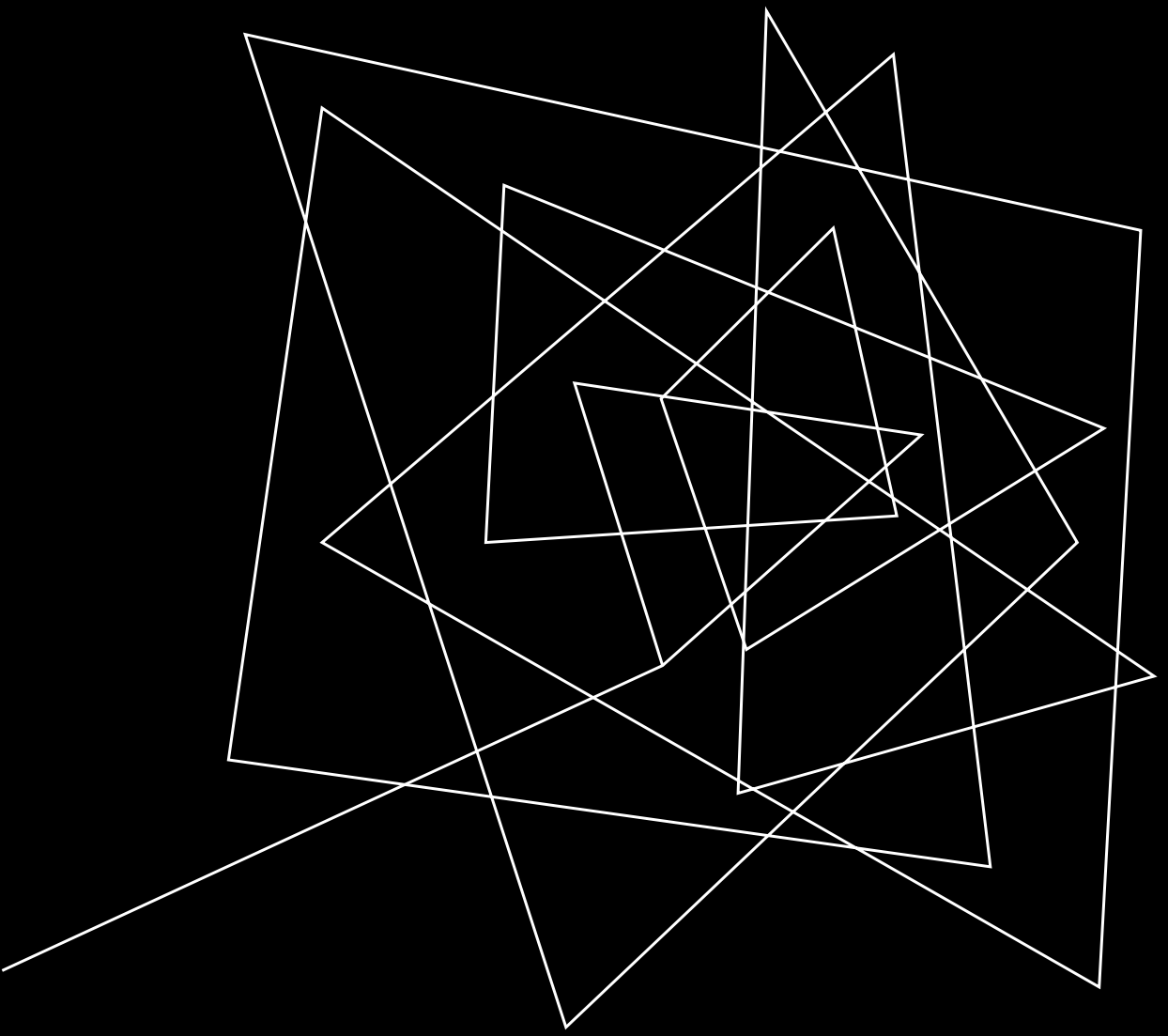
\home\chandra\nodejs-learnings\efficient_prog.js

WEB SERVER CONTD...

URL PARSING

- Object in NodeJS got properties and functions
- The ‘url’ module got one such object i.e., url which got properties and functions to do useful operations with URL
- ‘parse’ function gives us important properties, to write meaningful operations

\home\chandra\nodejs-learnings\url_parsing.js



MORE INTO NODEJS

OWN MODULES

- In NodeJS a file can contain useful functions to be used across
- Such file can be treated as module and can be exported to be used across

\home\chandra\nodejs-learnings\own_module.js

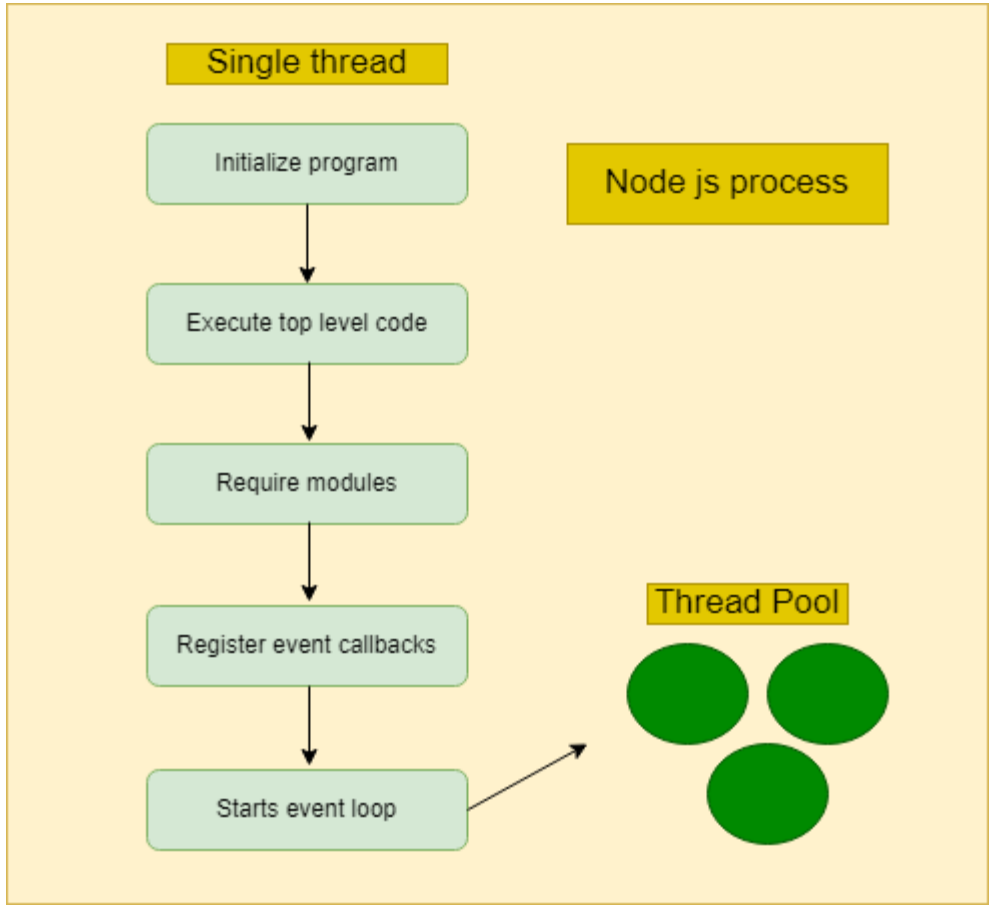
NODE PACKAGE MANAGER (NPM)

- Library/Registry for JavaScript software packages
- Got CLI, included in nodejs install, which helps to install different packages and manage their dependencies
- Command, `*npm init*`, creates package.json, which is configuration file for project that maintains packages information

mkdir npm_demo; cd npm_demo; npm init ; npm install slugify; npm install nodemon --save-dev; npm i nodemon --global

- nodemon \home\chandra\nodejs-learnings\first_prog.js

NODE JS PROCESS



- \home\chandra\nodejs-learnings\node_process.js

EVENT DRIVEN ARCHITECTURE

- Event is some notification, issues by someone. Listeners are the ones who listen to that event
- const server = http.createServer(); -> This command of starting server emits an event, *called as request*

nodemon \home\chandra\nodejs-learnings\http_events.js

nodemon \home\chandra\nodejs-learnings\event-emitter-demo.js

ASYNCHRONOUS CALLS – CLEANER WAY

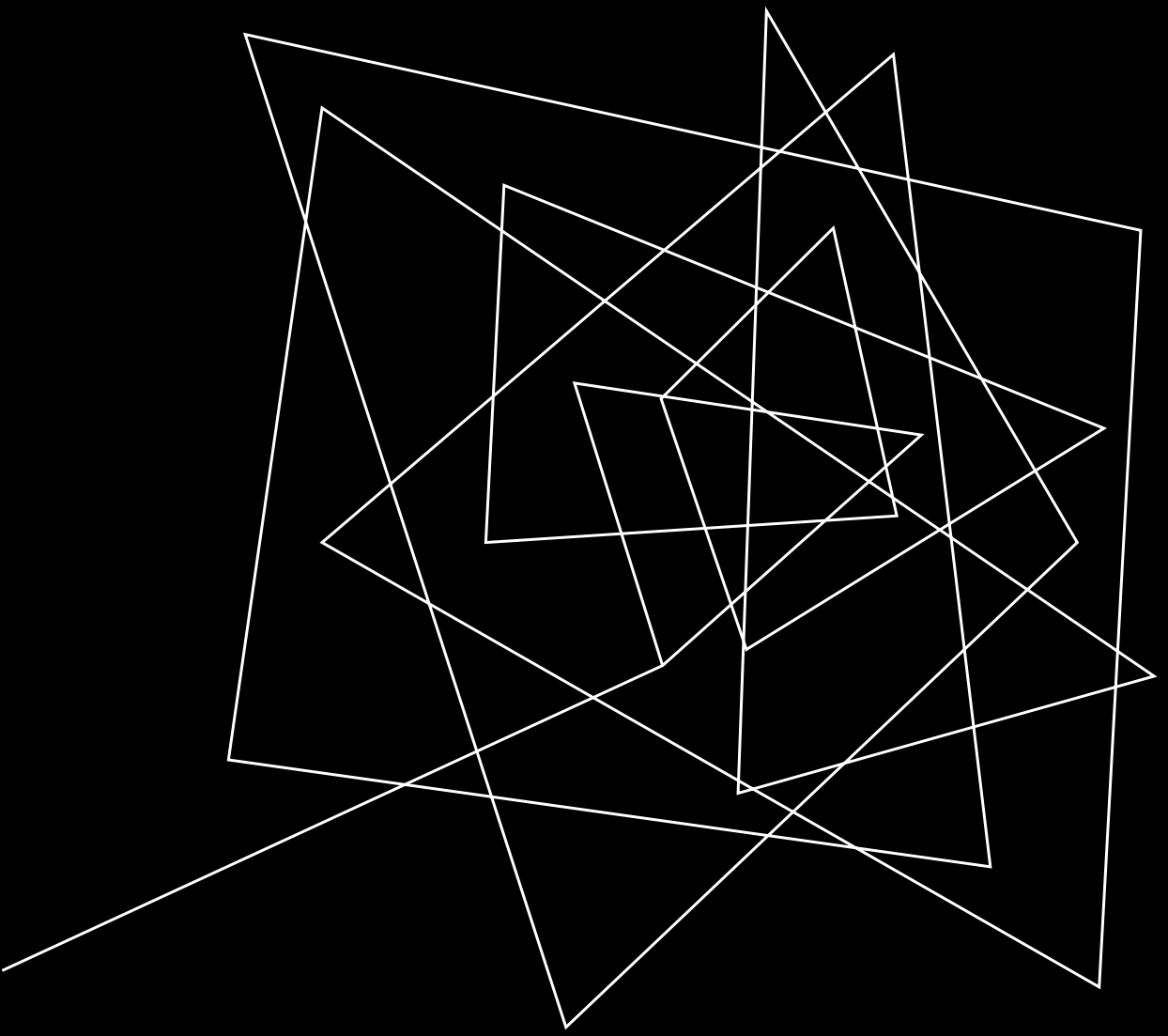
- Callback is one way , but bit messy
- *Promise* is cleaner way
- Promise is object, which is immediately available, and it promise us to return some data In future
- Promise can be in *pending* state, while data is not available
- Promise when gets data, is in *resolved* state
- Resolved promise might have error and thus can be in *rejected* state

nodejs-learnings\promise_demo\callback_hell.js
nodejs-learnings\promise_demo\read_promise.js
nodejs-learnings\promise_demo\write_promise.js
nodejs-learnings\promise_demo\promise_demo.js

ASYNC-AWAIT

- Mostly Promise would be consumed more than produced
- *async,* as the word suggest will process code in background allowing other things to be processed simultaneously
- *async* function will return *Promise (will see in demo)*
- Inside *async* function, there will be one or more *await* expressions
- *await* will wait till promise is resolved, so there is no need to '*then*' keyword (which again seen like callback hell)
- *await works only inside async function*
- *Its only blocks code inside async function, outside that method things proceed (so not same as blocking functions)*

nodejs-learnings\async_await_demo\asy_awa_pro_pend_demo.js
nodejs-learnings\async_await_demo\asy_awa_pro_res_demo.js
nodejs-learnings\async_await_demo\asy_awa_pro_avoid_then.js
nodejs-learnings\async_await_demo\asy_awa_pro_anon_way.js
nodejs-learnings\async_await_demo\asy_awa_pro_multi_promise.js



EXPRESS

EXPRESS

- Build on top of node js, makes easier to write node js applications

nodejs-learnings\express_demo\app_1.js (postman – GET app_1/, GET app_1/json, POST app_1)

REST APIS

- Express helps to quickly implement REST apis
- Demo for GET and POST request, with JSON object
- Express *middleware* are functions, which have access to request and response objects (*app demonstrate a usage to plug request json to middleware*)

nodejs-learnings\express_demo\app_2.js (postman – GET app_2/api/v1/fruits, GET app_2/api/v1/fruits/id, POST app_2/api/v1/fruits)

- Assignments – PUT and DELETE requests

REFACTORING - ROUTES

- Separate out routes from methods

nodejs-learnings\express_demo\app_3.js and
nodejs-learnings\express_demo\app_4.js (postman – GET app_2/api/v1/fruits, GET app_2/api/v1/fruits/id, POST app_2/api/v1/fruits) (postman requests for app_2 should work here)

- Assignments – PUT and DELETE requests

MIDDLEWARES

- Middleware function has access to request and response object
- Can execute any code
- Can make changes to request/response object
- Call next middleware in stack (must call next() to pass control to next middleware)
- Order matters, always get called in order

nodejs-learnings\express_demo\app_5.js (postman – GET app_2/api/v1/fruits, GET app_2/api/v1/fruits/id, POST app_2/api/v1/fruits) (postman requests for app_2 should work here) (additionally , GET app_2/api/v1/fruitspath/id for middleware demo to specific path)

- 'morgan' is middleware(3rd party), which logs the request with some other information

nodejs-learnings\express_demo\app_6.js (postman – GET app_2/api/v1/fruits, GET app_2/api/v1/fruits/id, POST app_2/api/v1/fruits) (postman requests for app_2 should work here)

USER DEFINED ROUTES

- Add few more routes

nodejs-learnings\express_demo\app_7.js (postman – GET app_2/api/v1/fruits, GET app_2/api/v1/users)

- This file is mess now , will all routes and methods at one place
- Let's Organize
- First create routers as middleware and mount those on paths

nodejs-learnings\express_demo\app_8.js (postman – GET app_2/api/v1/fruits, GET app_2/api/v1/users)

- Separate router functionality in diff files now

nodejs-learnings\express_demo\app_9.js (postman – GET app_2/api/v1/fruits, GET app_2/api/v1/fruits/id)

- The main app files looks like having middleware only

- Still route files got handlers. Let's separate those out in controller
- Let's also separate out server

nodejs-learnings\express_demo\server_1.js (postman – GET app_2/api/v1/fruits, GET app_2/api/v1/fruits/id)

REDEFINING FURTHER

- Have middleware (routerParam) , operational when id is supplied as parameter

nodejs-learnings\express_demo\server_2.js (postman – GET app_2/api/v1/fruits and pass param as 5)

- Clear separated logic here as routerParam is only defined for fruits and not for users
- Note addition of patch and delete methods in *fruit_controller_1.js*. These are incomplete methods, but the id check is seen repeated
- Fruit_controller_2.js is not having validation , exports.checkId

nodejs-learnings\express_demo\server_3.js (postman – GET app_2/api/v1/fruits and pass param as 51)

MIDDLEWARE CHAINING

- Middlewares can be chained, note exports.checkBody middleware which is chained in fruit_routes_5.js

nodejs-learnings\express_demo\server_4.js (postman – POST app_2/api/v1/fruits and ignore price or fruitName from body)

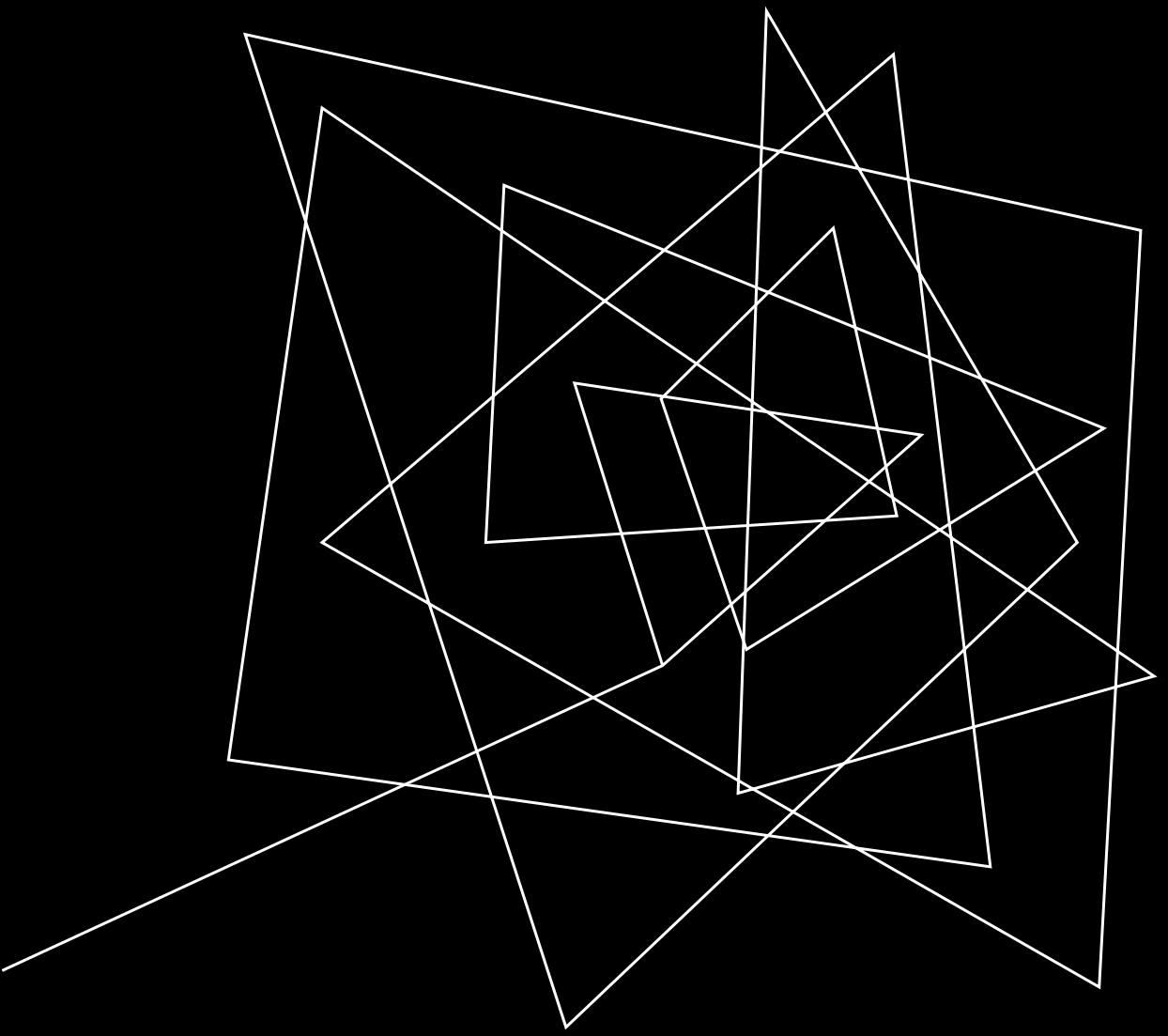
ENVIRONMENTAL VARIABLES

- The NODE_ENV is already defined for you and environmental variable can be passed via, command line

COURSE=DEV_OPS node nodejs-learnings\express_demo\server_5.js

- But defining on command line can only use for development purpose
- Instead, these should be placed in some configuration file
- Run - npm i dotenv

nodejs-learnings\express_demo\server_6.js (also note use of env variable in server_6.js)



MONGO

MONGO

- Free and Opensource
- Highly used DB with node
- Was developed by Atlas
- Google for atlas mongo, create account, create a free shared cluster, explore on your own, get connection string, user , password
- Put information in config.env

MONGOOSE

- npm i mongoose

COURSE=DEV_OPS node nodejs-learnings\express_demo\server_5.js

- Just check connection to mongo

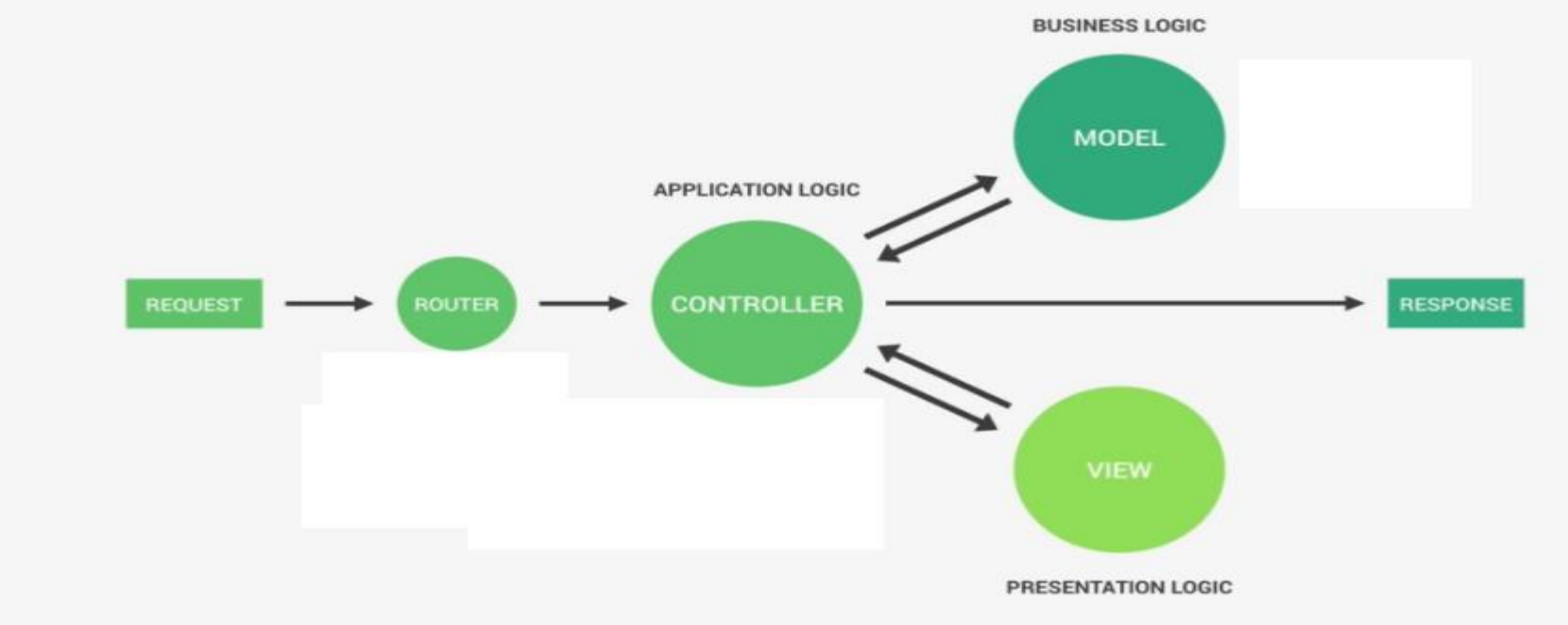
Nodemon nodejs-learnings\express_demo\server_7.js

MONGOOSE IN ACTION

- Schema defines data, datatypes, validate data, default values etc.,
- Models are created out of schema

nodejs-learnings\express_demo\server_8.js (verify on atlas)

MVC



- Model – Layer constitute of data and business logic
- Controller – Interact with business layer and send response back
- View – Display like web-pages

LET’S ORGANIZE IN CRUD OPERATIONS

- Let’s create model first and use it in controller

nodejs-learnings\express_demo\server_9.js (POST app_2/api/v1/fruits and check Atlas, GET app_2/api/v1/fruits)

- Verify and try other methods like patch , delete

BULK IMPORT

- Might require for initial data creation

```
nodejs-learnings\express_demo\import_data\import_data_data.js --import
nodejs-learnings\express_demo\import_data\import_data_data.js --delete
```

FURTHER LEARNING

Filtering – how to filter data when say price is greater than or less than

Atlas mongo features

Views with java script

Tool for mongo db, like Mongo Compass

SUMMARY

We learnt few features of node js, which enables us to design a MVC application where middleware interacts with backend.

We user atlas mongo db as our database, which is primarily used for node js

Always prefer node js for faster applications and not for CPU intensive applications