**CONTENTS** :

* Node concepts (Arch, Type of data, Server first calls, environment variables, Express, routing, middlewares, streams, event emitters, local system, web interactions)
* Starter kits
* Scaffolding
* NPM
* Web Server
* API Rest (level 2 & 3) concepts
* Transpile
* Bundling
* Linting
* Testing
* Debug
* Continuous Integration (Heroku)
* HTTP
* Build
* Deploy
* Best Practices
* Advanced concepts (Scaling, Sockets, Child process)
* GraphQL
* NoSQL databases (Dynamo, Mongo or another)
* JWT
* Courses
* Documentation
* Goodreads & Links
* Frameworks
* Introduction to Real App

**CLASS 1** :

**Folder Best practices** : <https://github.com/goldbergyoni/nodebestpractices#1-project-structure-practices>

**Middlewares** :

<https://expressjs.com/en/guide/using-middleware.html>

**Error Middlewares**:

Error-handling middleware always takes 4 arguments (err, req, res, next). You **must** provide four arguments to identify it as an error-handling middleware function. **Even if you don’t need to use the next object, you must specify it to maintain the signature**. Otherwise, the next object will be interpreted as regular middleware and will fail to handle errors.

app.use(function (err, req, res, next) {

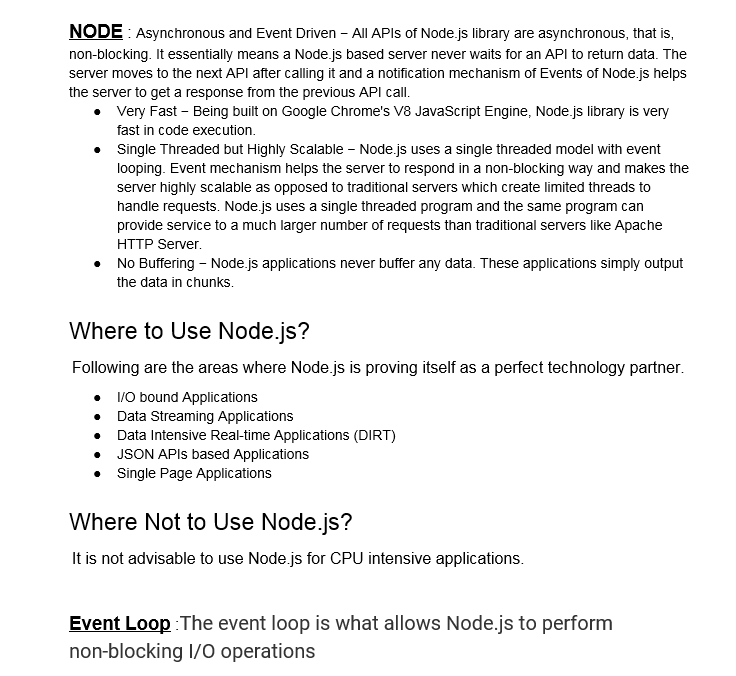
console.error(err.stack)

res.status(500).send('Something broke!')

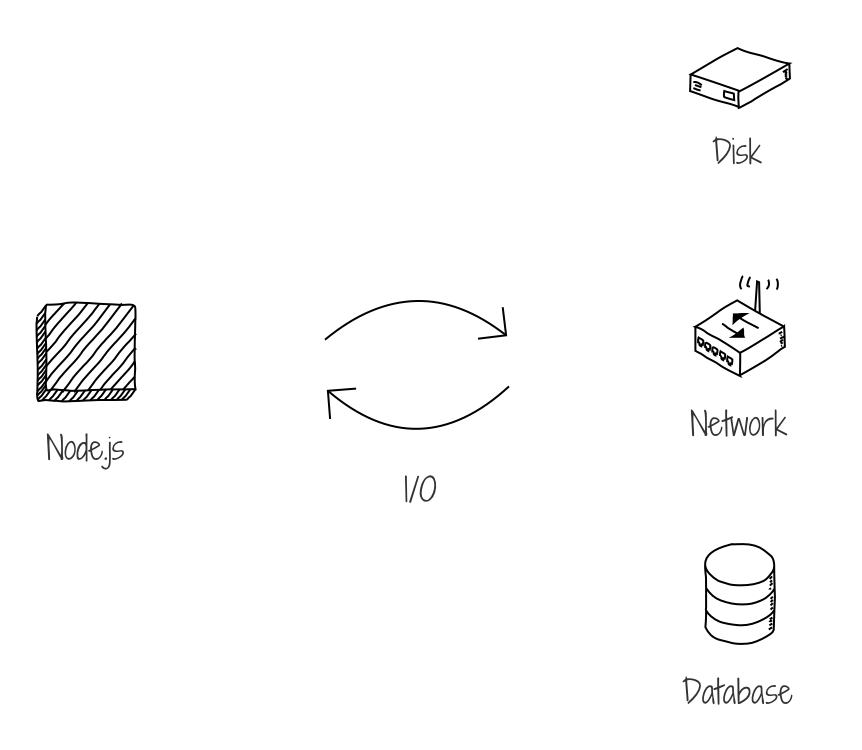
})

If you pass **anything** to the next() function (except the string 'route'), Express regards the current request as being **an error and will skip any remaining non-error handling routing** and middleware functions.

**CLASS 2** :



**I/O bound App** : Typical node app



**Data streaming App** : Streaming data is data that is continuously generated by different sources. Such data should be processed incrementally using Stream Processing techniques without having access to all of the data. In addition, it should be considered that concept drift may happen in the data which means that the properties of the stream may change over time. Eg: netflix, youtube, etc.

**Data Intensive App** : I/O intensive, large amounts of data or parallel processing

**JSON API** : Any json api is good example

**Single Page App** : Same as above

**CPU Intensive App** : mining (bitcoins) is a good example of cpu intensive app

**Event Loop** : <https://www.youtube.com/watch?v=8aGhZQkoFbQ>

**Advanced** :

Event loop -> phases

* NextTick()
* SetImmediate()

**CLASS 3 :**

**Plugins :**

[**https://www.npmjs.com/package/lint-staged**](https://www.npmjs.com/package/lint-staged)

[**https://www.npmjs.com/package/helmet**](https://www.npmjs.com/package/helmet)

**Eslint**

**"eslint-config-airbnb-base": "13.1.0",**

**"eslint-plugin-import": "2.16.0",**

**"eslint-plugin-mocha": "5.3.0",**

**"eslint-plugin-node": "8.0.1",**

**"eslint-plugin-security": "1.4.0",**

**"eslint-plugin-you-dont-need-lodash-underscore": "6.8.0",**

**.npmrc**

**npm config set save-exact=true**

**CLASS 4 :**

**Express best practices**

**Performance:** [**https://expressjs.com/en/advanced/best-practice-performance.html**](https://expressjs.com/en/advanced/best-practice-performance.html)

**Security:**

[**https://expressjs.com/en/advanced/best-practice-security.html**](https://expressjs.com/en/advanced/best-practice-security.html)

**Health-Check:**

[**https://expressjs.com/en/advanced/healthcheck-graceful-shutdown.html**](https://expressjs.com/en/advanced/healthcheck-graceful-shutdown.html)

**CLASS 5 :**

**API maturity levels**

[**https://blog.restcase.com/4-maturity-levels-of-rest-api-design/**](https://blog.restcase.com/4-maturity-levels-of-rest-api-design/)

**HTTP Status code sheet:**

****

**CLASS 6 :**

**Streams:** [**https://www.freecodecamp.org/news/node-js-streams-everything-you-need-to-know-c9141306be93/**](https://www.freecodecamp.org/news/node-js-streams-everything-you-need-to-know-c9141306be93/)

**Event Emitters:**

**File systems:**

[**https://nodejs.org/api/fs.html**](https://nodejs.org/api/fs.html)

**Web Interactions:**

[**https://nodejs.org/api/http.html**](https://nodejs.org/api/http.html)