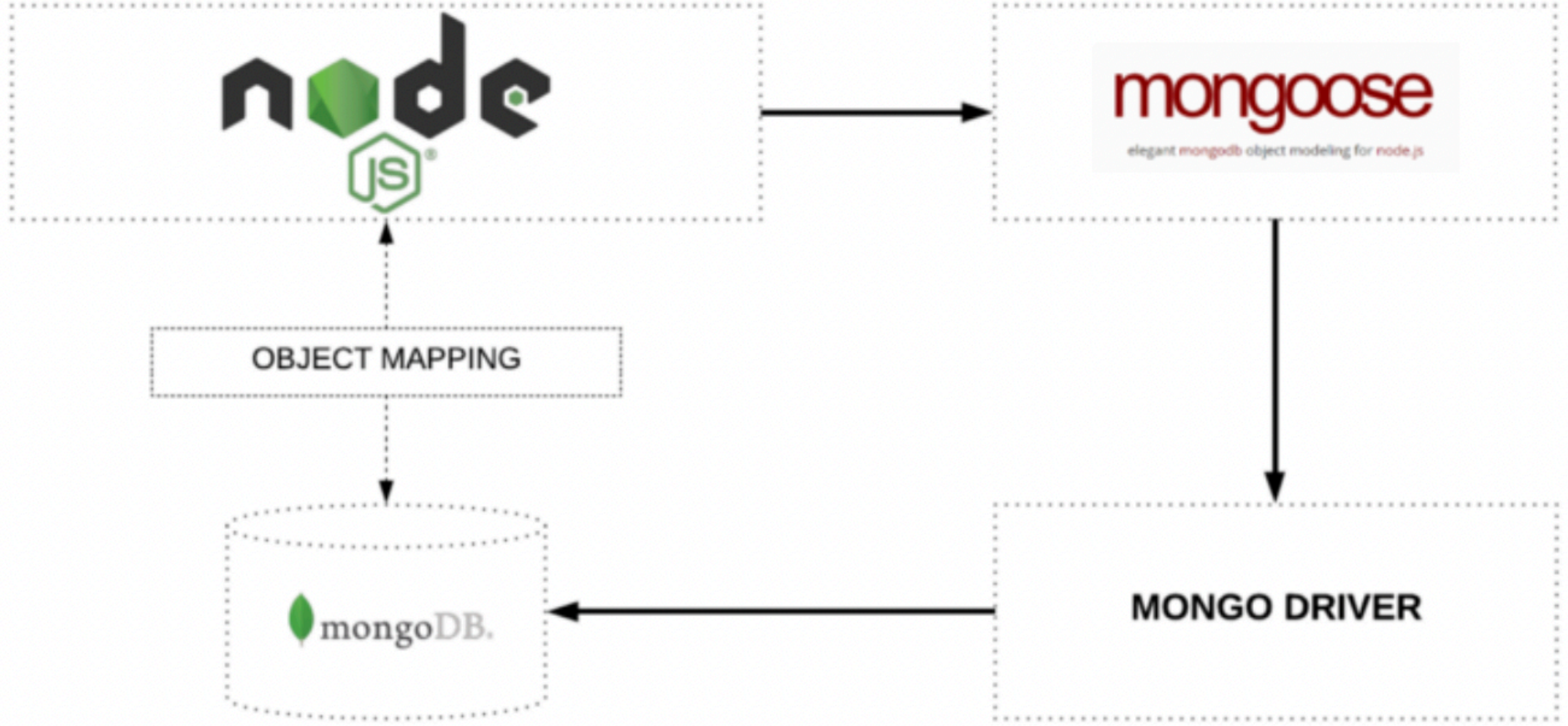
ANECO ACADEMY

NODEJS - MONGOOSE

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.



Object Mapping between Node and MongoDB managed via Mongoose

Collections

‘Collections’ in Mongo are equivalent to tables in relational databases. They can hold multiple JSON documents.

Documents

‘Documents’ are equivalent to records or rows of data in SQL. While a SQL row can reference data in other tables, Mongo documents usually combine that in a document.

Fields

‘Fields’ or attributes are similar to columns in a SQL table.

Schema

While Mongo is schema-less, SQL defines a schema via the table definition. A Mongoose ‘schema’ is a document data structure (or shape of the document) that is enforced via the application layer.

Models

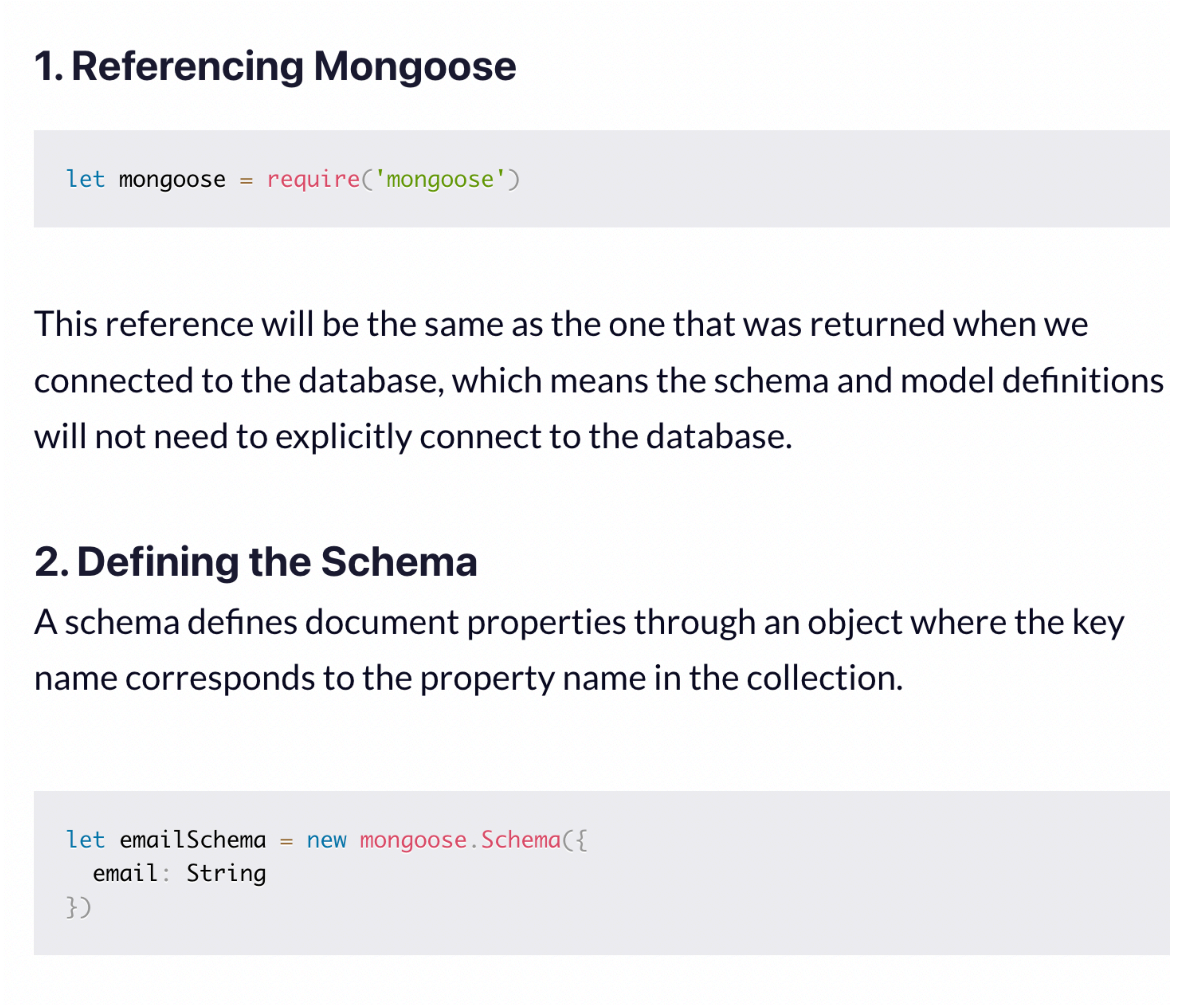
NPM PACKAGE

```
npm install mongoose
```

DATABASE CONNECTION

```
let mongoose = require('mongoose');

const server = '127.0.0.1:27017'; // REPLACE WITH YOUR DB SERVER
const database = 'fcc-Mail';      // REPLACE WITH YOUR DB NAME

class Database {
  constructor() {
    this._connect()
  }

_connect() {
     mongoose.connect(`mongodb://${server}/${database}`)
       .then(() => {
         console.log('Database connection successful')
       })
       .catch(err => {
         console.error('Database connection error')
       })
  }
}

module.exports = new Database()
```

SCHEMA DEFINE



1. Referencing Mongoose

```
let mongoose = require('mongoose')
```

This reference will be the same as the one that was returned when we connected to the database, which means the schema and model definitions will not need to explicitly connect to the database.

2. Defining the Schema

A schema defines document properties through an object where the key name corresponds to the property name in the collection.

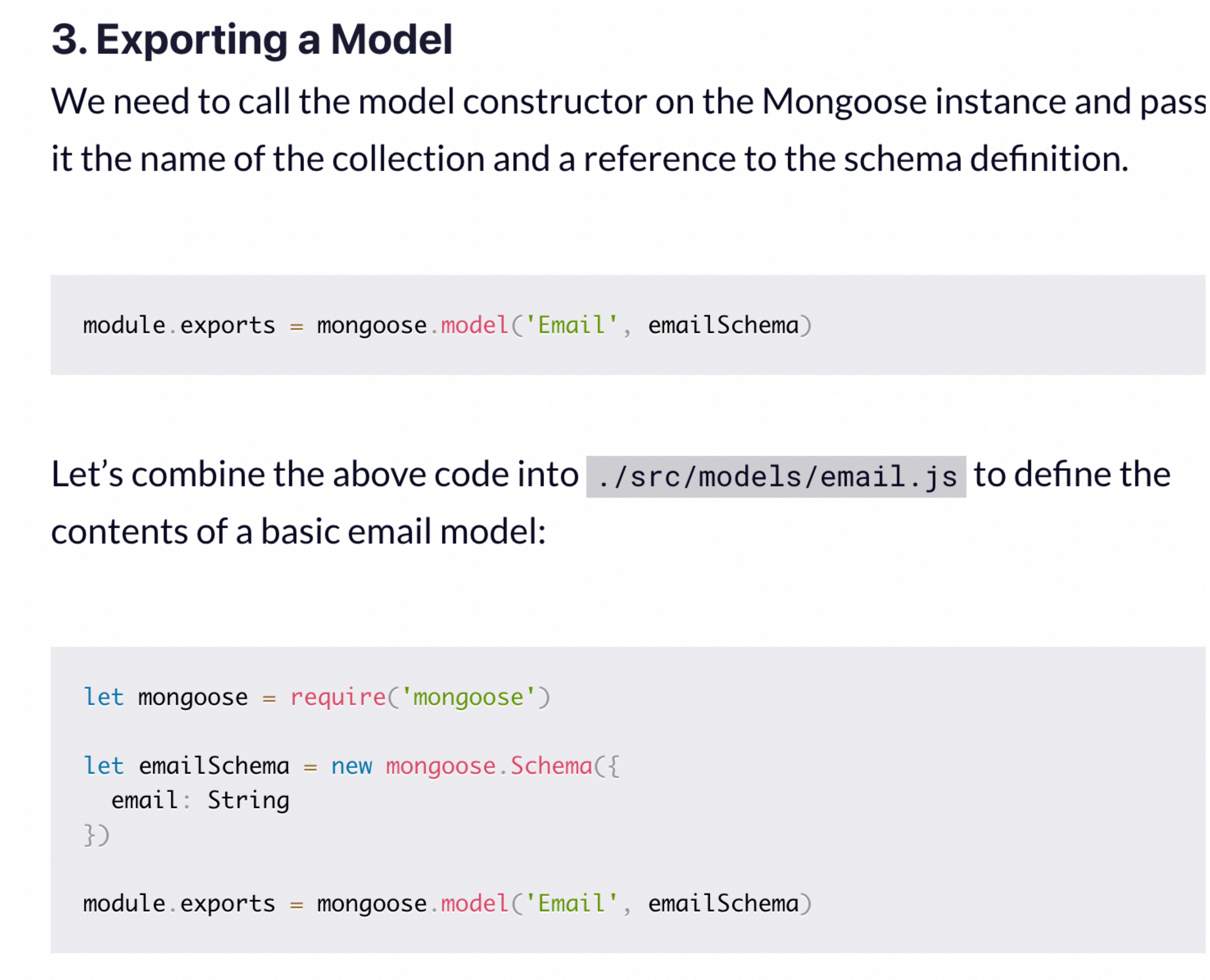
```
let emailSchema = new mongoose.Schema({
  email: String
})
```

DATATYPES

The following Schema Types are permitted:

- Array
- Boolean
- Buffer
- Date
- Mixed (A generic / flexible data type)
- Number
- ObjectId
- String

EXPORTING MODEL



3. Exporting a Model

We need to call the model constructor on the Mongoose instance and pass it the name of the collection and a reference to the schema definition.

```
module.exports = mongoose.model('Email', emailSchema)
```

Let’s combine the above code into `./src/models/email.js` to define the contents of a basic email model:

```
let mongoose = require('mongoose')

let emailSchema = new mongoose.Schema({
  email: String
})

module.exports = mongoose.model('Email', emailSchema)
```

SCHEMA STRUCTURE

```
let mongoose = require('mongoose')
let validator = require('validator')

let emailSchema = new mongoose.Schema({
  email: {
    type: String,
    required: true,
    unique: true,
    lowercase: true,
    validate: (value) => {
      return validator.isEmail(value)
    }
  }
})

module.exports = mongoose.model('Email', emailSchema)
```

SAVE METHOD

```
let EmailModel = require('./email')

let msg = new EmailModel({
  email: 'ADA.LOVELACE@GMAIL.COM'
})

msg.save()
   .then(doc => {
     console.log(doc)
   })
   .catch(err => {
     console.error(err)
   })
```

FIND METHOD

```
EmailModel
  .find({
    email: 'ada.lovelace@gmail.com'   // search query
  })
  .then(doc => {
    console.log(doc)
  })
  .catch(err => {
    console.error(err)
  })
```

UPDATE METHOD

```
EmailModel
  .findOneAndUpdate(
    {
      email: 'ada.lovelace@gmail.com'  // search query
    },
    {
      email: 'theoutlander@live.com'   // field:values to update
    },
    {
      new: true,                       // return updated doc
      runValidators: true              // validate before update
    })
  .then(doc => {
    console.log(doc)
  })
  .catch(err => {
    console.error(err)
  })
```

REMOVE METHOD

```
EmailModel
  .findOneAndRemove({
    email: 'theoutlander@live.com'
  })
  .then(response => {
    console.log(response)
  })
  .catch(err => {
    console.error(err)
  })
```