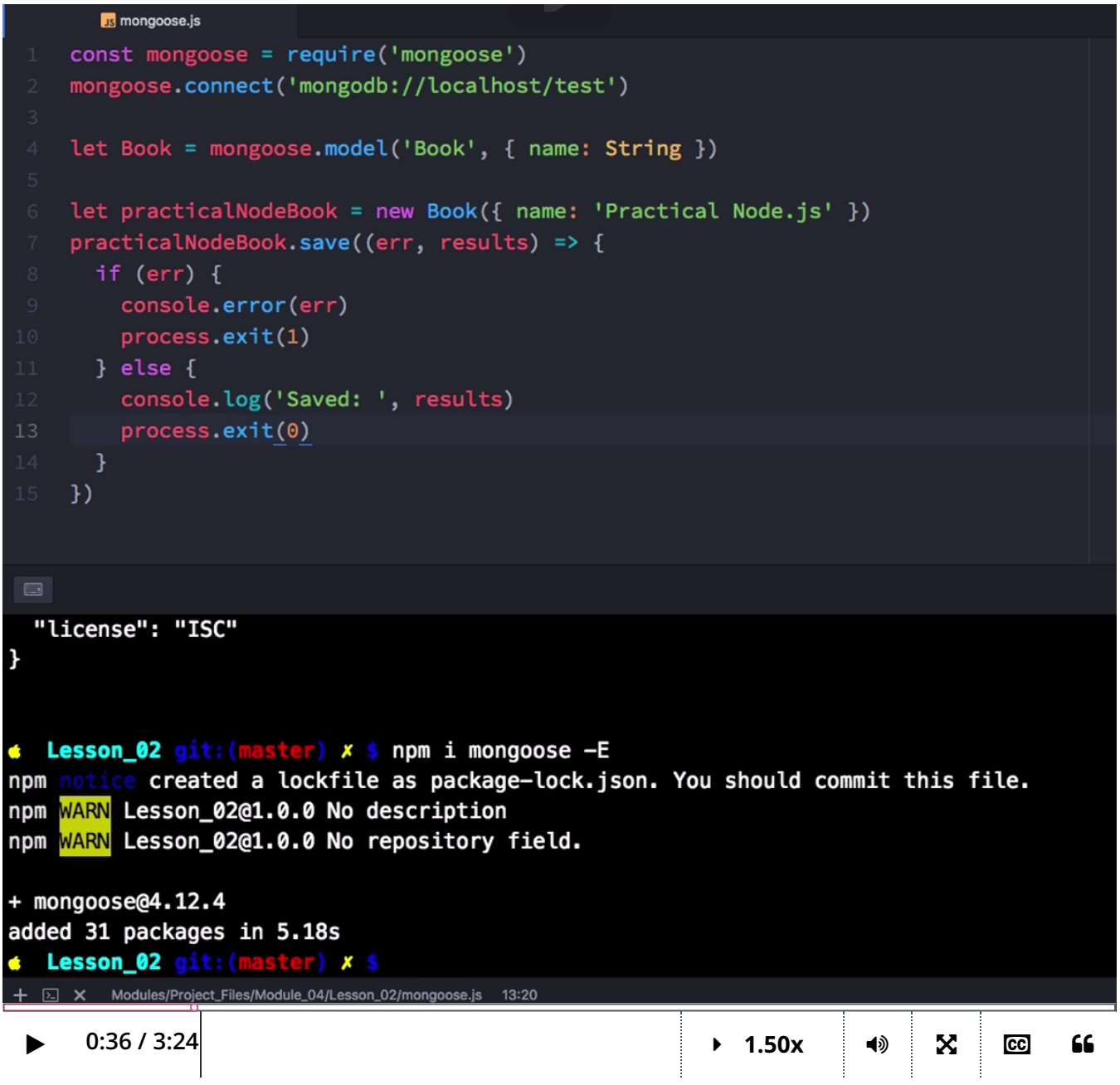
# Mongoose Basics
## Video: Mongoose Basics




## Video

Download video file

## Transcripts

**Download SubRip (.srt) file**

**Download Text (.txt) file**

---

# Mongoose Basics

It is straightforward to connect to MongoDB using Mongoose. To illustrate this, here's a simple script that establishes a connection, creates a Mongoose model definition, instantiates the `practicalNodeBook` object, and then saves it to the database.

To have access to the library, we need to include the mongoose module in our program:

```
const mongoose = require('mongoose')
```

Unlike the Node.js native MongoDB driver, which requires us to write a few lines of code, Mongoose can connect to the database server in one line. Mongoose requests are buffered, so we don't have to wait for the established connection (vs. the MongoDB native driver, which usually requires a callback). To do this, just call `mongoose.connect(uri(s), [options], [callback])`.

The uniform resource identifier (URI) or connection string is the only required parameter, and it follows a standard format of `type://username:password@host:port/database_name`. In our simple example, the host is `localhost`, the port is 27017 (default), and the database name is `test`:

```
mongoose.connect('mongodb://localhost/test')
```

Now the configuration phase is over and we can create a document that represents a particular instance of the model `Book`:

```
let Book = mongoose.model('Book', { name: String })
let practicalNodeBook = new Book({ name: 'Practical Node.js' })
```

Mongoose documents come with very convenient built-in methods (http://mongoosejs.com/docs/api.html#document-js) such as `validate`, `isNew`, `update`, and so on. Just keep in mind that these methods apply to this particular document, not the entire collection or model. The difference between documents and models is that a document is an instance of a model; a model is something abstract.

It's like your real MongoDB collection, but it is supported by a schema and is presented as a Node.js class with extra methods and attributes. Collections in Mongoose closely resemble collections in Mongoskin or native driver. Strictly speaking, models, collections, and documents are different Mongoose classes.

Usually we don't use Mongoose collections directly, and we manipulate data via models only. Some of the main model methods look strikingly familiar to the ones from Mongoskin or native MongoDB driver, such as `find`, `insert()`, `save`, and so forth.

To finish our small script and make it write a document to the database, let's use one of the document methods—`document.save()`:

```
practicalNodeBook.save((err, results) => {
  if (err) {
    console.error(err)
    process.exit(1)
  } else {
    console.log('Saved: ', results)
    process.exit(0)
  }
})
```

Here is the full source code for the `mongoose.js` file:

```
const mongoose = require('mongoose')
mongoose.connect('mongodb://localhost/test')

let Book = mongoose.model('Book', { name: String })

let practicalNodeBook = new Book({ name: 'Practical Node.js' });
practicalNodeBook.save((err, results) => {
  if (err) {
    console.error(err)
    process.exit(1)
  } else {
    console.log('Saved: ', results)
    process.exit(0)
  }
})
```

To run this snippet, execute the `node mongoose.js` command (MongoDB server must be running in parallel).