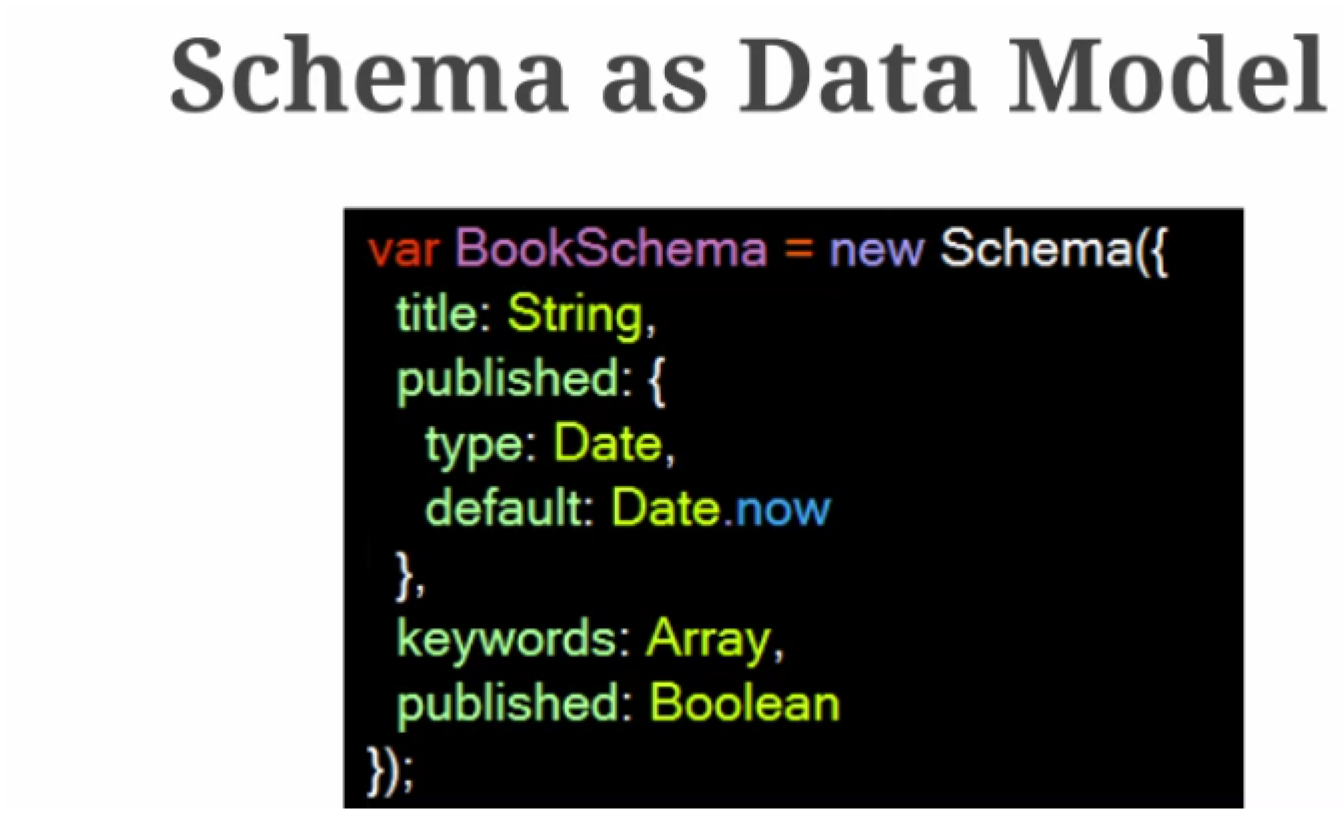
Mongoose JS

An Overview

Schema as Data Model

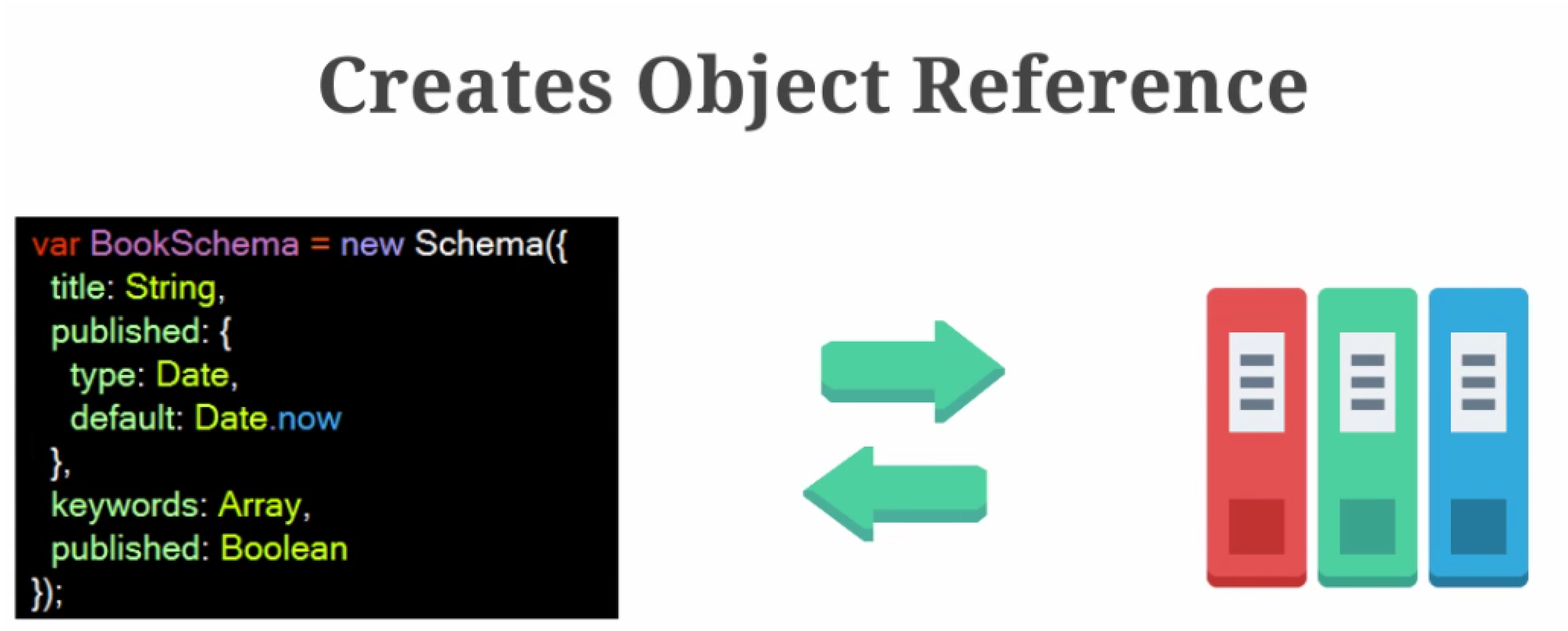
How to use?



Schema as Data Model

```
var BookSchema = new Schema({
 title: String,
 published: {
   type: Date,
   default: Date.now
 },
 keywords: Array,
 published: Boolean
});
```

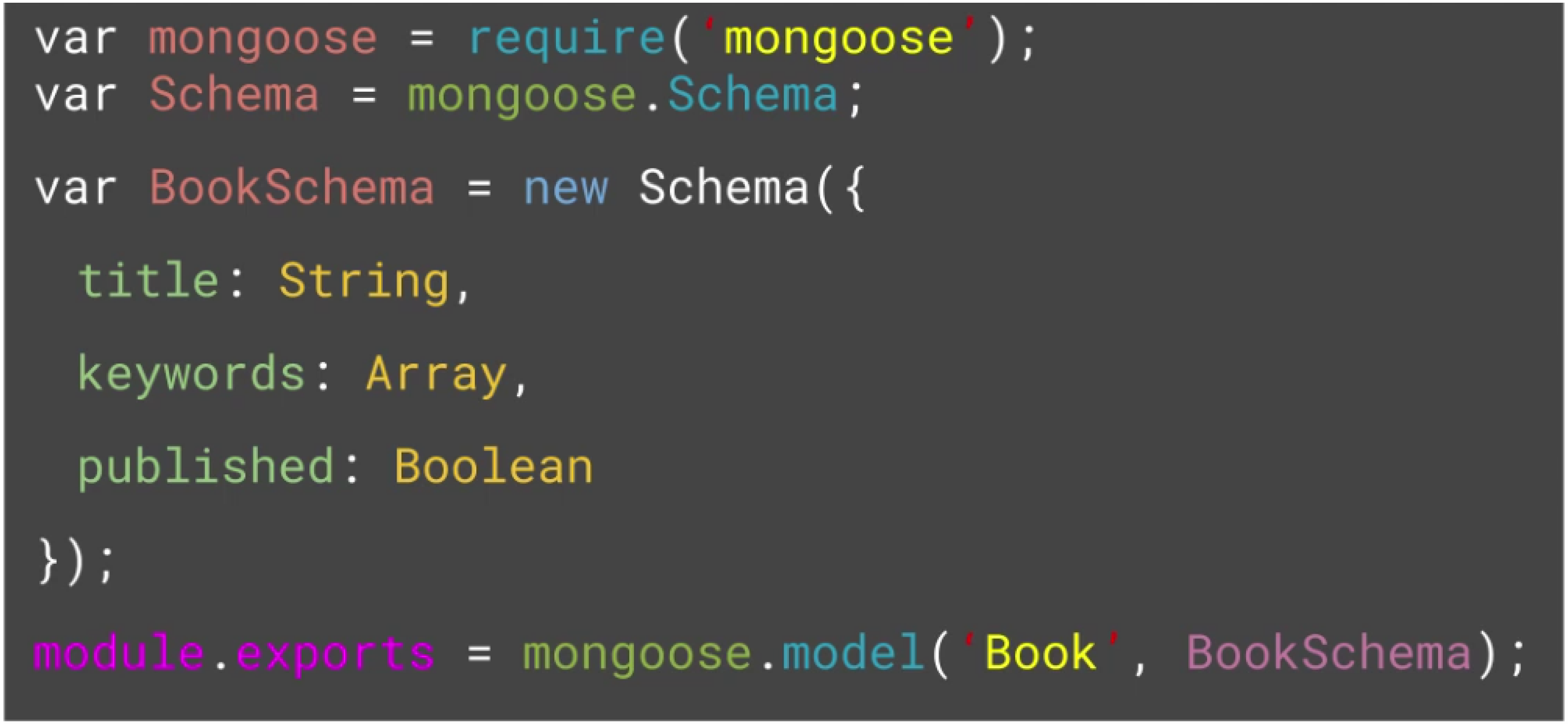
What is does?



Creates Object Reference

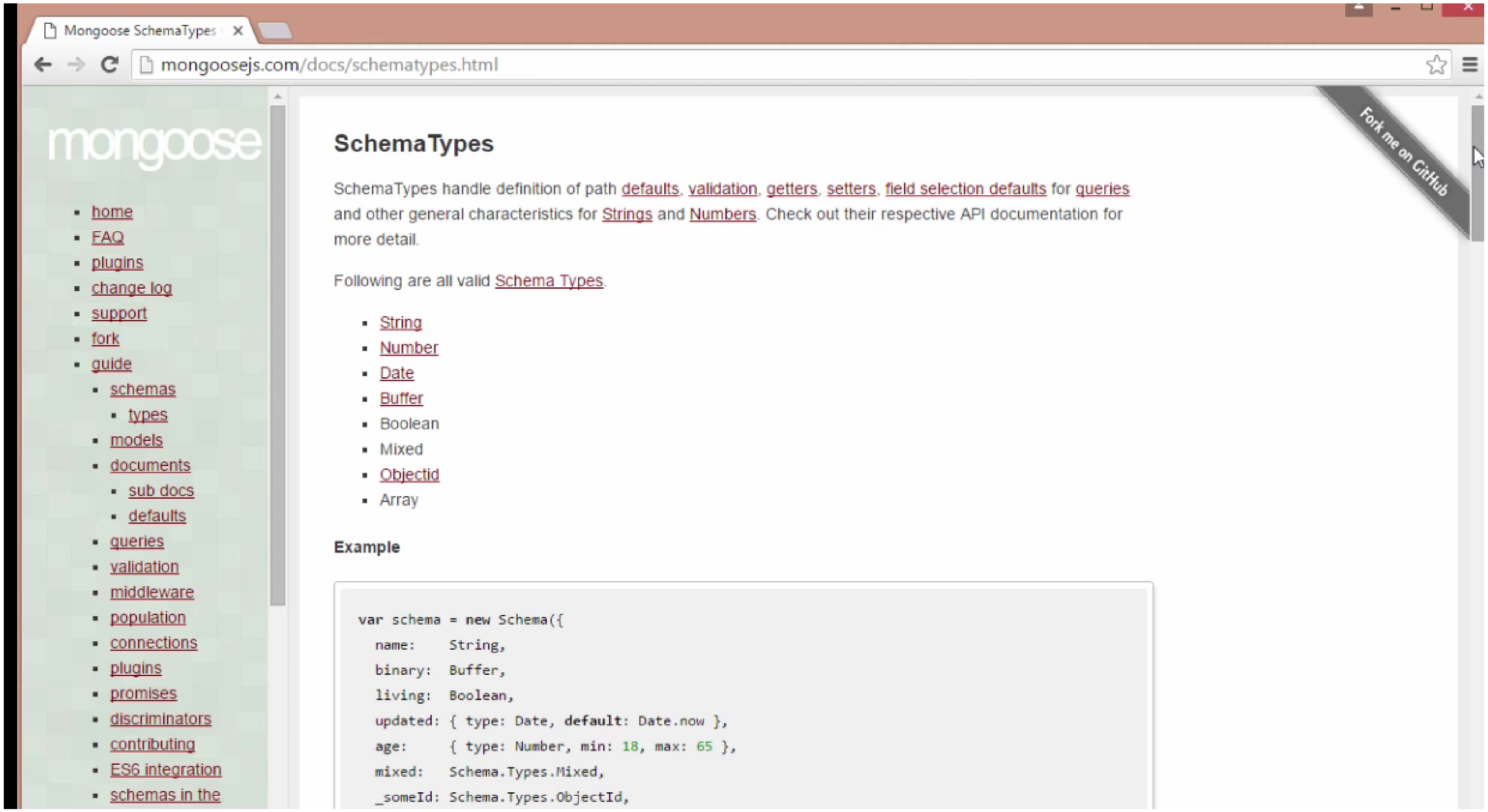
```
var BookSchema = new Schema({
  title: String,
  published: {
    type: Date,
    default: Date.now
  },
  keywords: Array,
  published: Boolean
});
```

How to use?



```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;

var BookSchema = new Schema({

  title: String,

  keywords: Array,

  published: Boolean

});

module.exports = mongoose.model('Book', BookSchema);
```

Where to find more information?


Mongoose SchemaTypes

mongoosejs.com/docs/schematypes.html

- home
- FAQ
- plugins
- change log
- support
- fork
- guide
  - schemas
    - types
  - models
  - documents
    - sub docs
    - defaults
  - queries
  - validation
  - middleware
  - population
  - connections
  - plugins
  - promises
  - discriminators
  - contributing
  - ES6 integration
  - schemas in the

SchemaTypes

SchemaTypes handle definition of path defaults, validation, getters, setters, field selection defaults for queries and other general characteristics for Strings and Numbers. Check out their respective API documentation for more detail.

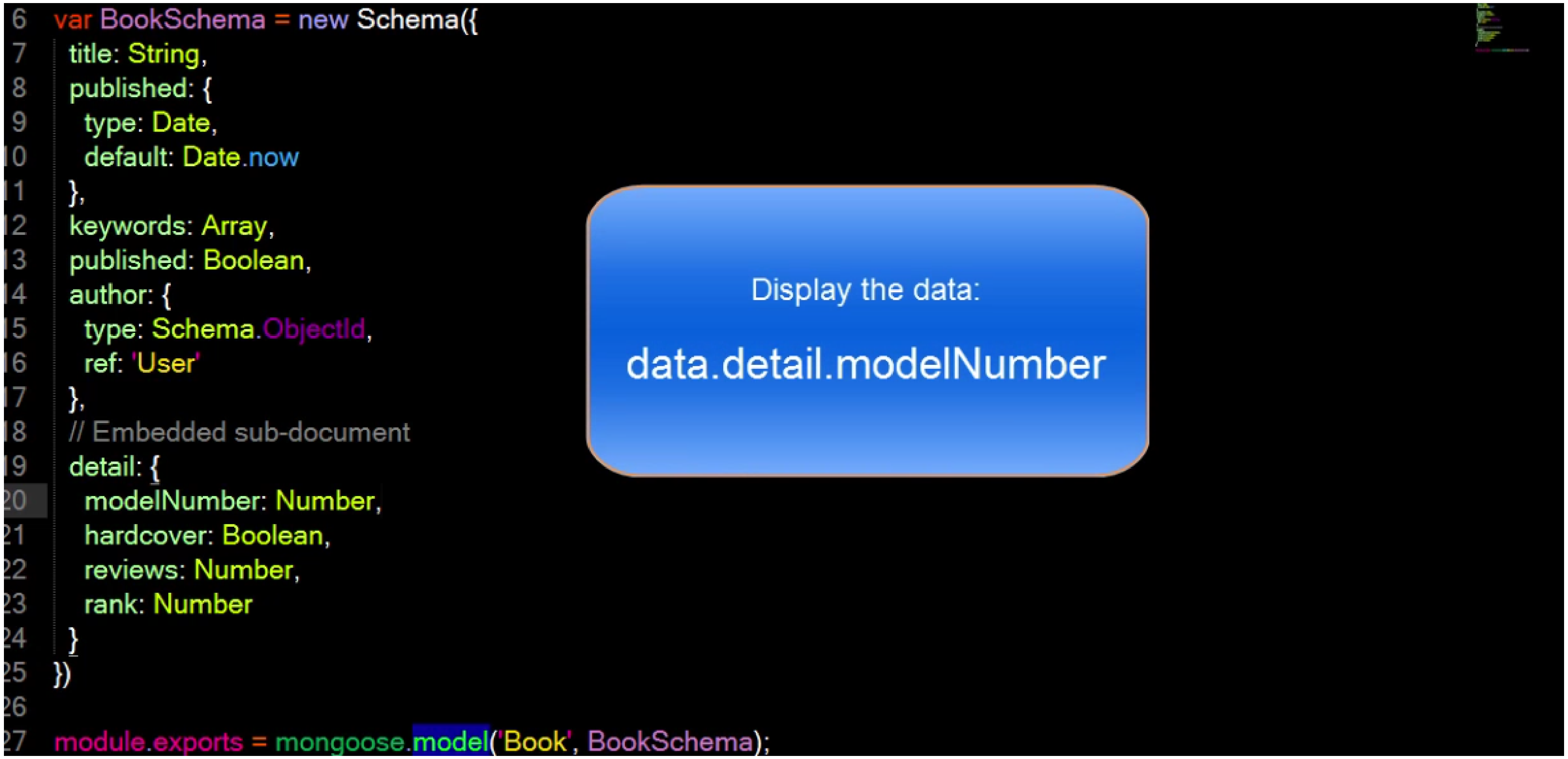
Following are all valid Schema Types.

- String
- Number
- Date
- Buffer
- Boolean
- Mixed
- Objectid
- Array

Example

```
var schema = new Schema({
  name:    String,
  binary:  Buffer,
  living:  Boolean,
  updated: { type: Date, default: Date.now },
  age:     { type: Number, min: 18, max: 65 },
  mixed:   Schema.Types.Mixed,
  _someId: Schema.Types.ObjectId,
```

How it helps?

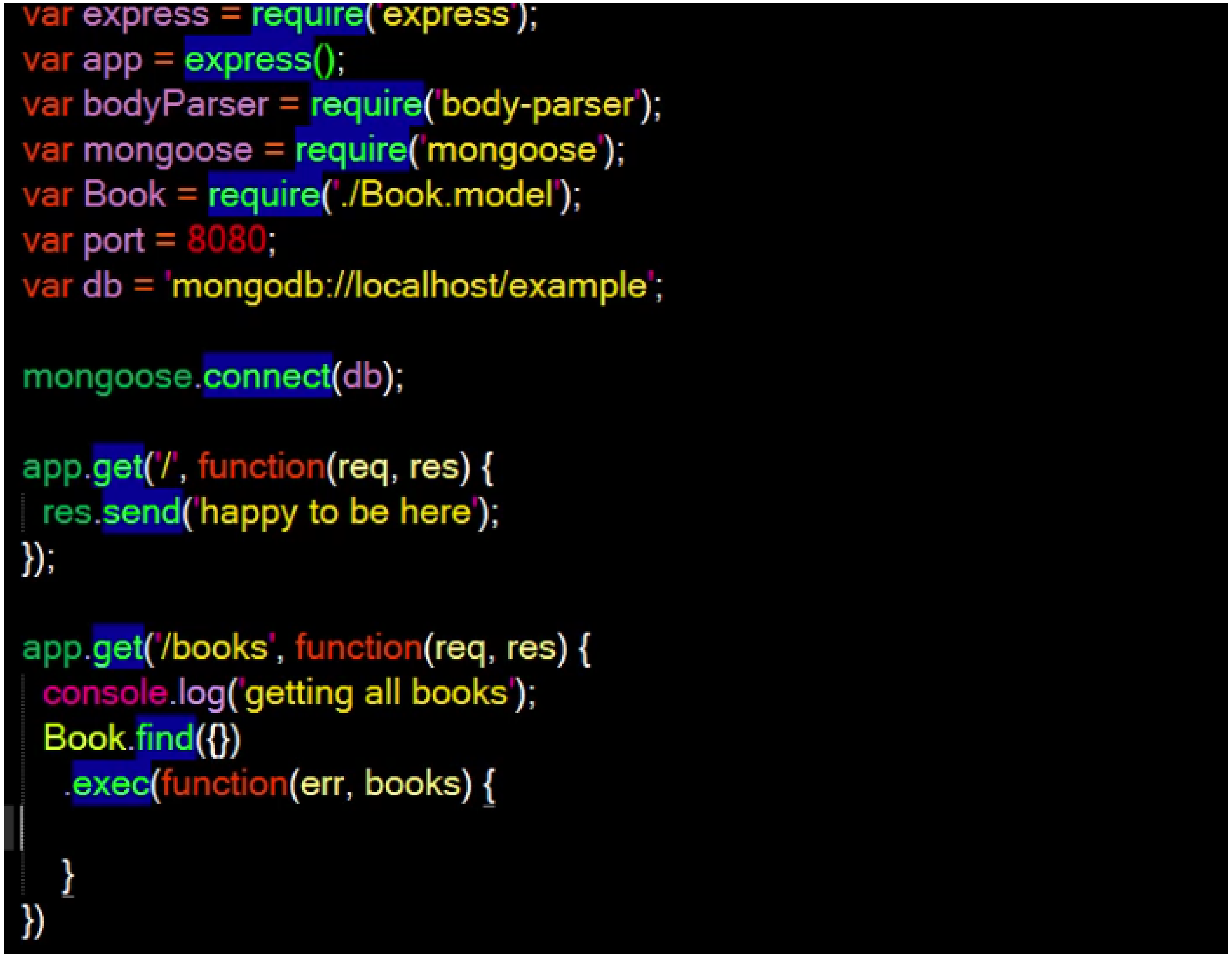


```
var BookSchema = new Schema({
  title: String,
  published: {
    type: Date,
    default: Date.now
  },
  keywords: Array,
  published: Boolean,
  author: {
    type: Schema.ObjectId,
    ref: 'User'
  },
  // Embedded sub-document
  detail: {
    modelNumber: Number,
    hardcover: Boolean,
    reviews: Number,
    rank: Number
  }
})

module.exports = mongoose.model('Book', BookSchema);
```

Display the data:

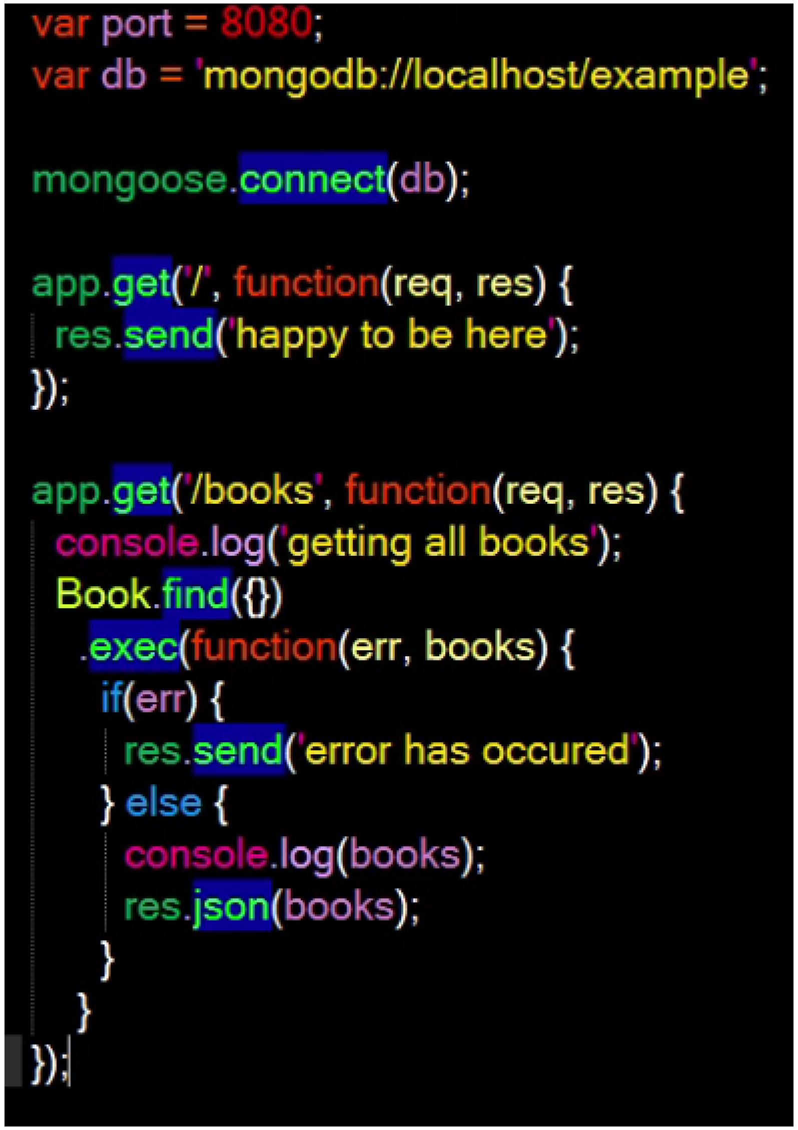
data.detail.modelNumber

findAll()



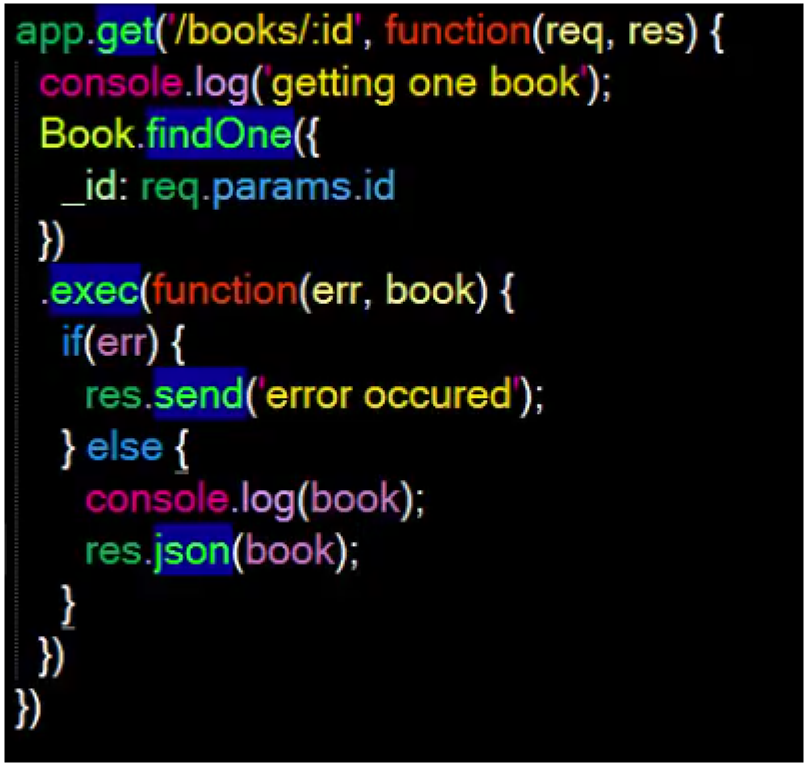
```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
var mongoose = require('mongoose');
var Book = require('./Book.model');
var port = 8080;
var db = 'mongodb://localhost/example';

mongoose.connect(db);

app.get('/', function(req, res) {
  res.send('happy to be here');
});

app.get('/books', function(req, res) {
  console.log('getting all books');
  Book.find({})
    .exec(function(err, books) {

    }
})
```

findAll()



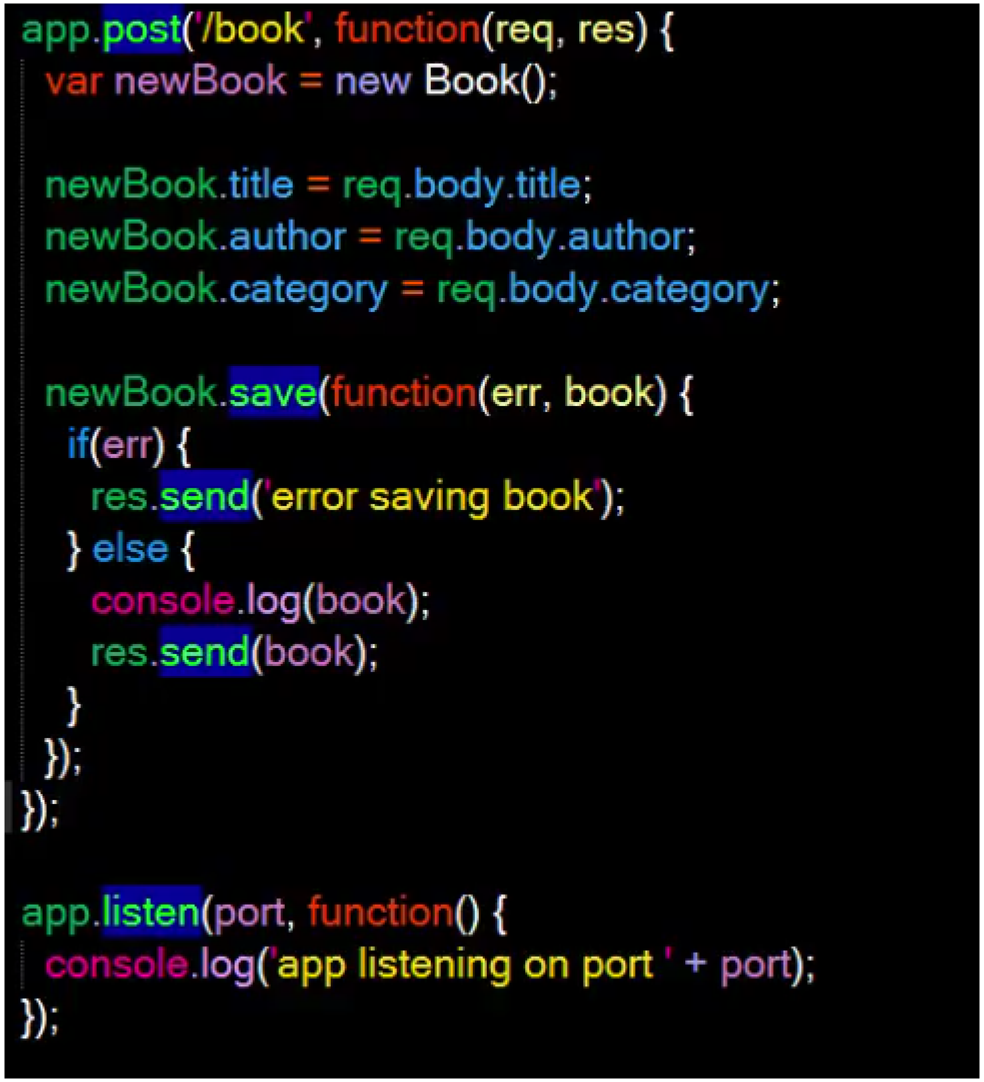
```
var port = 8080;
var db = 'mongodb://localhost/example';

mongoose.connect(db);

app.get('/', function(req, res) {
  res.send('happy to be here');
});

app.get('/books', function(req, res) {
  console.log('getting all books');
  Book.find({})
    .exec(function(err, books) {
      if(err) {
        res.send('error has occured');
      } else {
        console.log(books);
        res.json(books);
      }
    }
});
```

findOne()

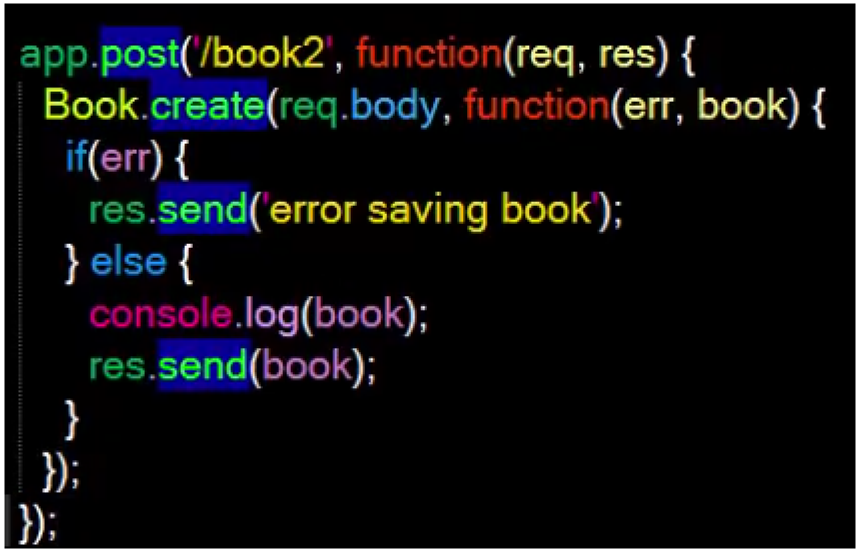


```
app.get('/books/:id', function(req, res) {
  console.log('getting one book');
  Book.findOne({
    _id: req.params.id
  })
  .exec(function(err, book) {
    if(err) {
      res.send('error occured');
    } else {
      console.log(book);
      res.json(book);
    }
  })
})
```

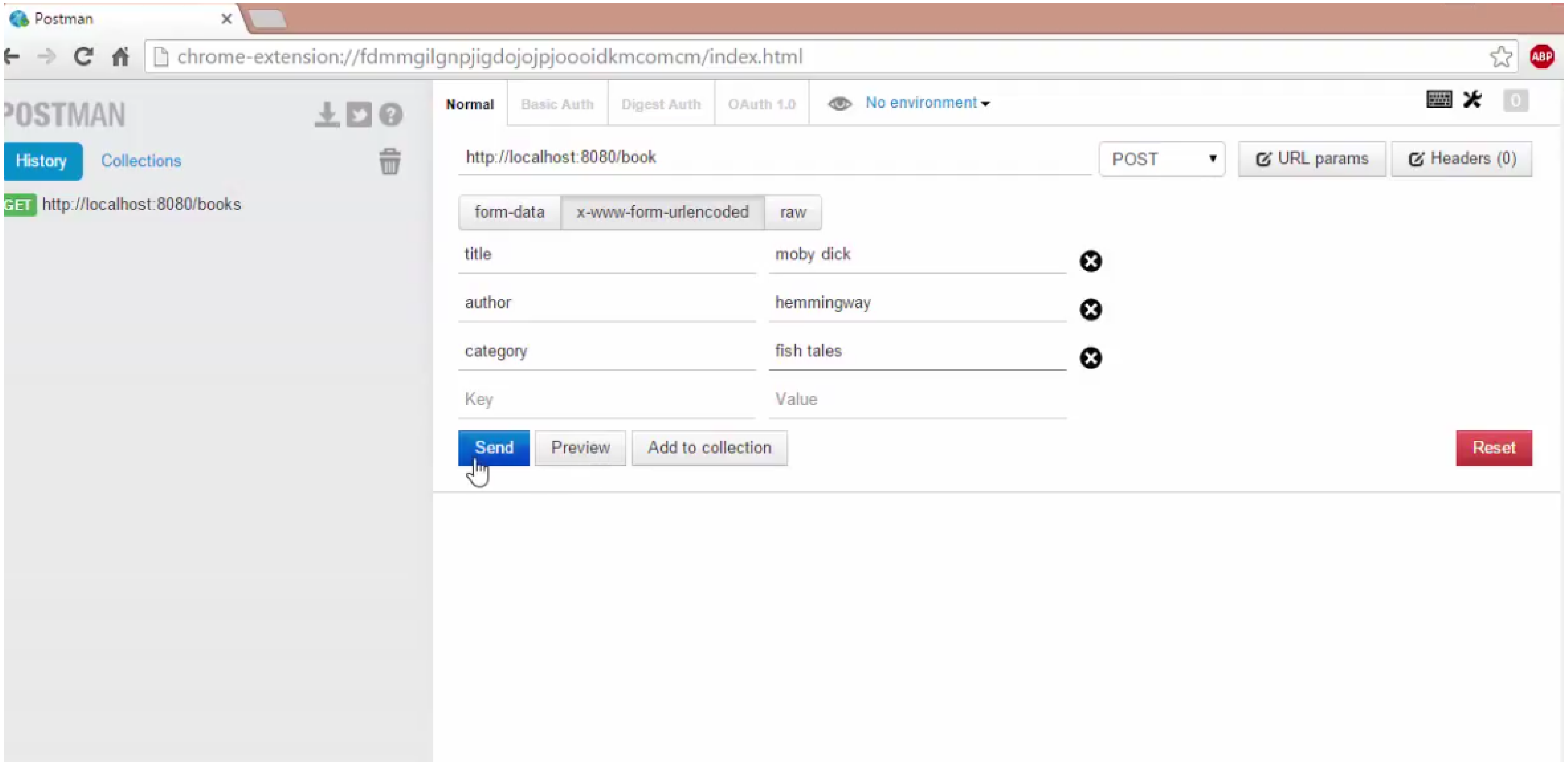
Save() aka update()



```
app.post('/book', function(req, res) {
  var newBook = new Book();

  newBook.title = req.body.title;
  newBook.author = req.body.author;
  newBook.category = req.body.category;

  newBook.save(function(err, book) {
    if(err) {
      res.send('error saving book');
    } else {
      console.log(book);
      res.send(book);
    }
  });
});

app.listen(port, function() {
  console.log('app listening on port ' + port);
});
```



```
app.post('/book2', function(req, res) {
  Book.create(req.body, function(err, book) {
    if(err) {
      res.send('error saving book');
    } else {
      console.log(book);
      res.send(book);
    }
  });
});
```



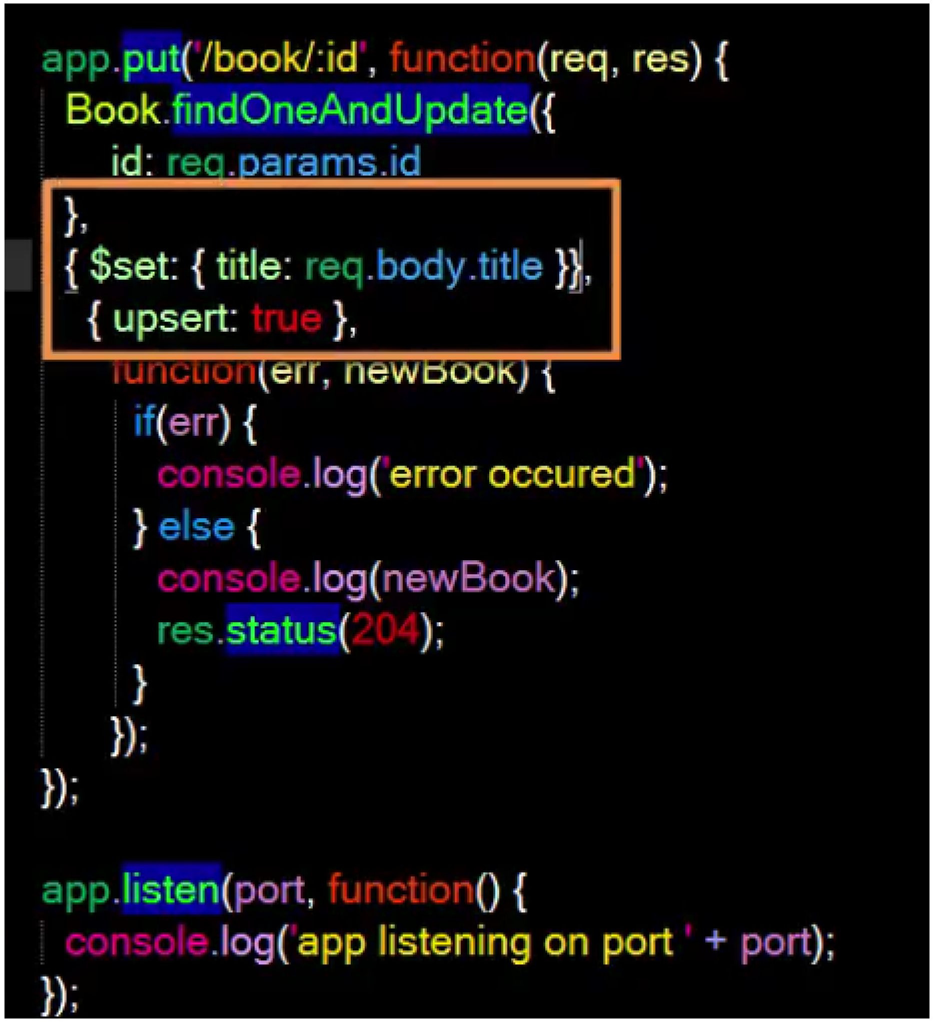
GET http://localhost:8080/books

http://localhost:8080/book POST

[form-data ✓ x-www-form-urlencoded raw]

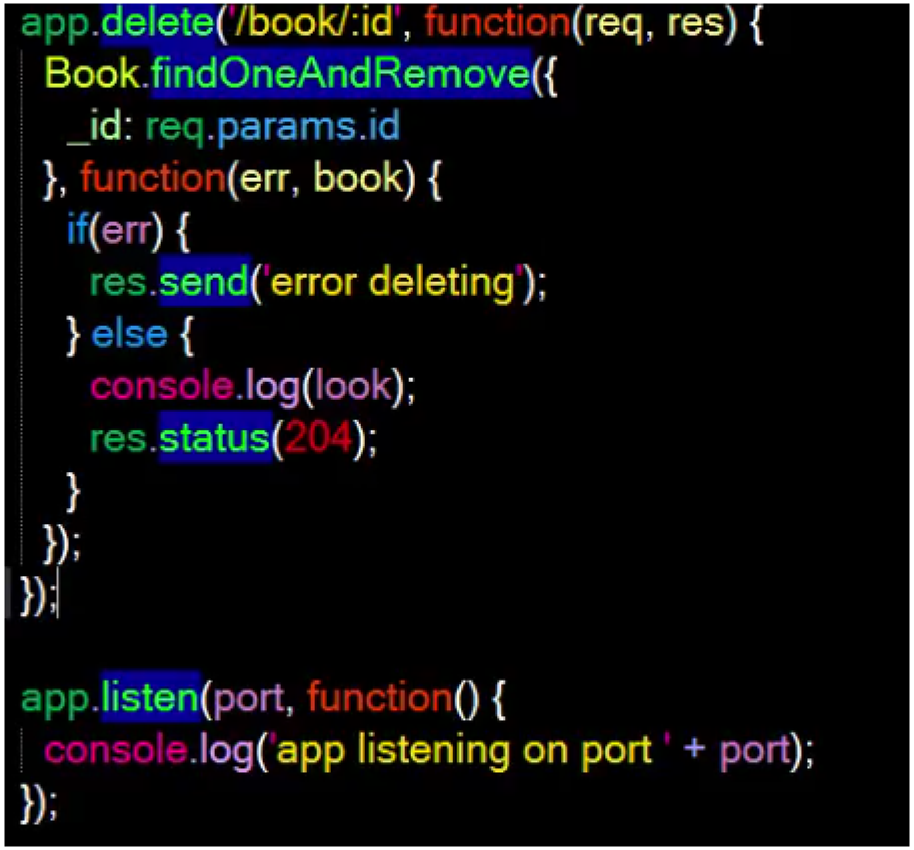
title	moby dick
author	hemmingway
category	fish tales
Key	Value

update()



```
app.put('/book/:id', function(req, res) {
  Book.findOneAndUpdate({
      id: req.params.id
  },
  { $set: { title: req.body.title }},
    { upsert: true },
      function(err, newBook) {
        if(err) {
          console.log('error occured');
        } else {
          console.log(newBook);
          res.status(204);
        }
      });
});

app.listen(port, function() {
  console.log('app listening on port ' + port);
});
```

delete()



```
app.delete('/book/:id', function(req, res) {
  Book.findOneAndRemove({
    _id: req.params.id
  }, function(err, book) {
    if(err) {
      res.send('error deleting');
    } else {
      console.log(look);
      res.status(204);
    }
  });
});

app.listen(port, function() {
  console.log('app listening on port ' + port);
});
```