Case study - The Contacts Web API

- Design:
  - Get http://localhost:4000/ - index.html (from public folder) + assets.
  - Get http://localhost:4000/api/contacts - Get all contacts
  - Put http://localhost:4000/api/contacts/:id - Update specific contact. Ex.:

    PUT http://localhost:4000/api/contacts/54fb37dfc350189c3f949bce
  - Delete http://localhost:4000/api/contacts/:id - Delete a contact

Case study – The Hacker News Web API

- Design:
  - Get /api/posts – Get all posts
  - Get /api/posts/:id – Get a specific post
  - Post /api/posts – Add a new post
  - Post /api/posts/:id/upvotes – Change a post's upvote count.
  - Post /api/posts/:id/comments – Add a new comment to a post.
  - Post /api/posts/:post_id/comments/:comment_id/upvotes – Change the upvote count for a comment

Mongoose

Alternative to native driver,

Simplifies MongoDB interfacing.

.

Problem.

- **MongoDB collections are not like SQL tables.**
- **Documents can vary in structure.**
- **How to handle validation without structure?**

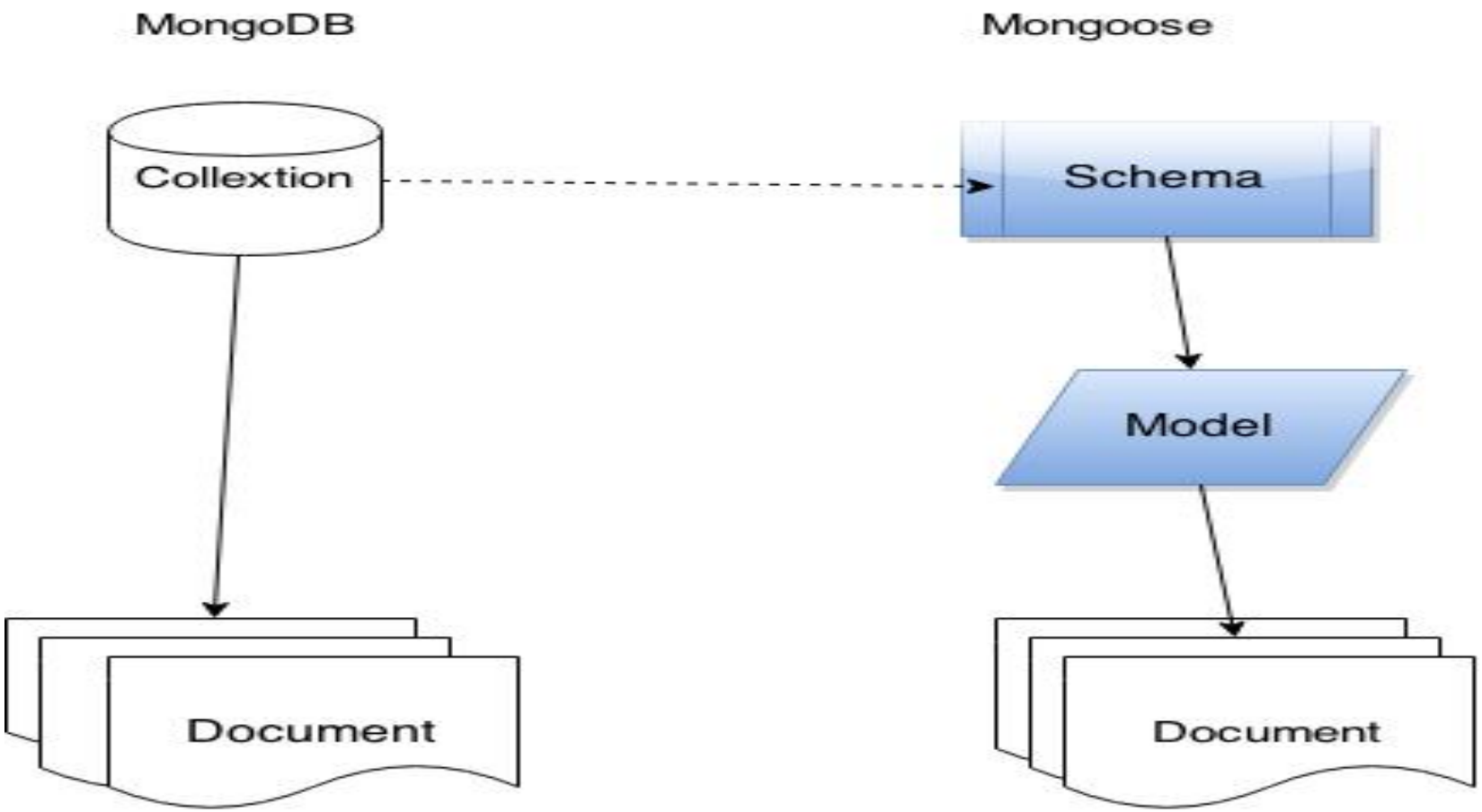
  Or

- **How to handle structure?**

Mongoose.

- *MongoDB Object modeling* for node.js.
- Provides two things:
    1. Schemas, and
    2. Validation.

The concepts.



Example – The Contacts collection

```
1  var mongoose = require('mongoose');
2  mongoose.connect('mongodb://localhost/contactsDB');
3
4  var Schema = mongoose.Schema   // Constructor function
5  var ContactSchema = new Schema({
6    name: { type: String, required: true },
7    address: String,    // Default is optional
8    phone_number: { type: String, required: true}
9  });
10 // Associate ContactSchema with the contacts collection
11 var ContactModel = mongoose.model('contacts', ContactSchema);
12
```

- **For the contacts database collection the schema instance (ContactSchema) declares:**
  1. **A Structure, and**
  2. **A set of Validation constraints (required:true)**

Basic Query

- **We interact with the database collection via the model.**

```
13  // Query collection for ALL documents
14  ContactModel.find(function (err, contacts) {
15      if(err) {
16        console.log(err);
17      } else {
18        contacts.forEach(function(contact) {
19            console.log(contact.name)
20            })
21      }
22      process.exit()
23   })
24  console.log('I appear immediately')
```

  - **find( )**
    - **get all (or get some)**
    - **A non-blocking method.**

Mongoose with Express

- **Mongoose is more typically used in a Web API context rather than in a standalone application.**

```
 9  // Get list of contacts
10  exports.index = function(req, res) {
11    Contact.find(function (err, contacts) {
12      if(err) { return handleError(res, err);
13      return res.json(200, contacts);
14    });
15  } ;
```

  - **handleError() – local function that returns a 500 HTTP status code**

Insert document.

```
26  var newContact = { name: 'Phil Jones',
27                     address: '2 Main St',
28                     phone_number: '022-123123'}
29  ContactModel.create(newContact , function(err, contact) {
30      if(err) {
31          console.log(err)
32      } else {
33        console.log (contact._id);
34      }
35      process.exit()
36  });
```

- **create()**
  - **Fails if newContact violates any validation constraint declared in related schema.**

    { name: 'Phil Jones', phone_number: '022-123123'} [Success]

    { name: 'Phil Jones', address: '2 Main St' } [Error]

    { name: 'Phil Jones', phone_number: '022-123123', age: 23}

    [Success, but age is not saved to the database]

Update document.

```
41  ContactModel.findOne({name: 'Pat Smith' } , function (err, contact) {
42        if (err) {
43            console.log(err)
44        } else {
45            contact.address = '5 Main St'
46            contact.save(function (err) {
47                if (err) {
48                    console.log(err);
49                } else {
50                    console.log (' Address updated')
51                }
52                process.exit()
53            });
54         }
55  });
```

- save( ) - method of a document object previously read from the database

Basic Query.

- **find() versus findOne( )**
  - **One returns an array of documents, the other returns a single document (the first match)**

- **ContactModel.findOne({name: 'Pat Smith' } , function (err,foo) {**
  - **foo is a document object**
  - **foo.save() will work**

- **ContactModel.find({name: 'Pat Smith' } , function (err,foo) {**
  - **foo is an array of document objects.**
  - **foo.save() will not work, but, foo[0].save() will work.**

Basic Query.

```
57  var theID = "550c435eeac6e7ebc3eaed37"
58
59  ContactModel.findById(theID , function (err, contact) {
60      if (err) {
61          console.log(err)
62      } else {
63          console.log(contact.name)
64      }
65      process.exit()
66  });
```

- **findById( )**
  - **Id based queries are always quicker than findOne()**

Alternative Query syntax.

```
68  var query = ContactModel.find()
69  // do other stuff
70  query.exec( function (err, contacts) {
71      if(err) {
72        console.log(err);
73      } else {
74        contacts.forEach(function(contact) {
75            console.log(contact.name)
76            })
77      }
78      process.exit()
79   })
80
```

- This syntax allows us 'build' more complex queries, then execute them.

Complex Query syntax.

- **Example: Find all contacts aged between 17 and 66, living in either Waterford, Kilkenny or Wexford.**

```
81  // sample contact = {name: ... , address: ..., age: 24,
82  //                   phone_number: ..., county: 'Waterford'}
83
84  var query = ContactModel
85      .where('age').gt(17).lt(66)
86      .where('county').in(['Waterford', 'Kilkenny','Wexford'])
87
88  query.exec( function (err, contacts) { ...} )
89
```

- **Based on query meta-properties of MongoDB query syntax - $gt, $gte, $in, $nin, etc**

Complex Query syntax.

- **A query can also:**
    1. **Limit the result set size, e.g. first 10 matches.**
    2. **Sort the result set, e.g. ascending name order**
    3. **Select a subset of the document properties, e.g. name and phone number properties only.**

  - **Any mixture allowed, e.g. 1 and 3, 2 and 3.**

Complex Query syntax.

- **Example: Get all contacts and sort in descending name order; only read names and phone number properties.**

```
93  var query = ContactModel.find()
94              .sort('-name')
95              .select('name phone_number')
96
97  query.exec( function (err, contacts) {
98      if(err) {
99        console.log(err);
100     } else {
101       contacts.forEach(function(contact) {
102           console.log(contact)
103           })
104     }
105     process.exit()
106  })
```

```
$ node mongoose_demo.js
{ name: 'Phil Jones',
  phone_number: '022-123123',
  _id: 550d481ebf1859eec727a352 }
{ name: 'Pat Smith',
  phone_number: '011-123123',
  _id: 550c435eeac6e7ebc3eaed37 }
{ name: 'Jane Fleming',
  phone_number: '321-45678',
  _id: 54fb37dfc350189c3f949bce }
{ _id: 54fb39bcc350189c3f949bcf,
  name: 'Jane Collins',
  phone_number: '321-45678123' }
$
```

Complex Query syntax.

- **Example: The first 10 contacts aged between 17 and 66 from Waterford, Kilkenny or Wexford, sorted in ascending name order – name and phone number properties only required .**

```
98  var query = ContactModel
99      .where('age').gt(17).lt(66)
100     .where('county').in(['Waterford', 'Kilkenny','Wexford'])
101     .limit(10)
102     .sort('+name')   // As
103     .select('name phone_number')
104
105 query.exec( function (err, contacts) { ...} )
106
```

Schemas.

- **Schema types**
  - **String,**
  - **Number,**
  - **Boolean,**
  - **Date,**
  - **Mixed,**
  - **Array.**

- **Sample on the right:**
  - **comments property is an array of sub-documents or embedded documents**
  - **meta property is a single sub-document**

```
var blogSchema = new Schema({
  title:  String,
  author: String,
  body:   String,
  comments: [{ body: String, date: Date }],
  date: { type: Date, default: Date.now },
  hidden: Boolean,
  meta: {
    votes: Number,
    favs:  Number
  }
});
```

Validation.

- **Validation is defined in the SchemaType.**
- **Validation occurs when a document attempts to be saved.**
- **Validation is asynchronously recursive; when you call Document#save, sub-document validation is executed as well. If an error occurs, your Documentl#save callback receives it**
- **Validation supports complete customization.**

Validation.

- **Several built in validators.**
  - **All SchemaTypes have the required validator.**
  - **Numbers have min and max validators.**
  - **Strings have enum and match validators.**

```
106  var Contact2Schema = new Schema({
107    name: { type: String, required: true },
108    county: {
109          type: String,
110          enum: ['TY', 'KK', 'WD', 'WX'],
111          required: true
112      },
113    address: String,
114    age: { type: Number, min: 5, max: 110 },
115    phone_number: { type: String, required: true}
116  });
117  var Contact2Model = mongoose.model('contacts2', Contact2Schema);
118
```

The Error object (1/2).

- **The document below falis many validation constraints:**
  1. **Phone number is missing**
  2. **Age is below the minimum**
  3. **XX is not in the county set**

```
119  var newContact2 = { name: 'Philip Jones',
120                      county: 'XX',
121                      age: 2,
122                      address: '2 Main St'
123                    }
124
125  Contact2Model.create(newContact2 , function(err, contact) {
126      if(err) {
127          console.log(err)
128       }
129      process.exit()
130  });
131
```

The Error object (2/2).

```
$ node mongoose_demo.js
{ [ValidationError: Validation failed]
  message: 'Validation failed',
  name: 'ValidationError',
  errors:
   { phone_number:
      { [ValidatorError: Path `phone_number` is required.]
        message: 'Path `phone_number` is required.',
        name: 'ValidatorError',
        path: 'phone_number',
        type: 'required',
        value: undefined },
     age:
      { [ValidatorError: Path `age` (2) is less than minimum allowed value (5).]
        message: 'Path `age` (2) is less than minimum allowed value (5).',
        name: 'ValidatorError',
        path: 'age',
        type: 'min',
        value: 2 },
     county:
      { [ValidatorError: `XX` is not a valid enum value for path `county`.]
        message: '`XX` is not a valid enum value for path `county`.',
        name: 'ValidatorError',
        path: 'county',
        type: 'enum',
        value: 'XX' } } }
$
```

Custom validator (1/2)

- EX: A contact's address must be between 5 and 40 characters in length.

```
132 Contact2Schema.path('address').validate(function (v) {
133     if (v.length > 40 || v.length < 5) {
134       return false
135     }
136     return true
137 }, 'Contact address should be between 5 and 40 characters');
138
139 var newContact2_2 = { name: 'Philip Jones',
140                       county: 'KK',
141                       address: 'home',
142                       phone_number: '123-456789'
143                     }
144
145 Contact2Model.create(newContact2_2 , function(err, contact) {...
150 });
151
```

Custom validator (2/2)

```
$ node mongoose_demo.js
{ [ValidationError: Validation failed]
  message: 'Validation failed',
  name: 'ValidationError',
  errors:
   { address:
      { [ValidatorError: Contact address should be between 5 and 40 characters]
        message: 'Contact address should be between 5 and 40 characters',
        name: 'ValidatorError',
        path: 'address',
        type: 'user defined',
        value: 'home' } } }
$
```

Sub-documents (Embedded documents)

- EX: A post document has an array of comment sub-documents.

```
152  var CommentSchema = new Schema({
153      body: { type: String, required: true},
154      author: { type: String, required: true },
155      upvotes: Number
156    });
157
158  var PostSchema = new Schema({
159    title: { type: String, required: true },
160    link: { type: String, optional: true },
161    username: { type: String, optional: true },
162    comments: [CommentSchema],
163    upvotes: { type: Number, min: 0 , max:100}
164  });
165
166  var Post = mongoose.model('posts', PostSchema);
```

Adding Sub-documents

```
194  Post.findById('5510117c1a9f03cd1ed38a6c', function (err,p){
195          if (err) {
196            console.log (err)
197          } else {
198              p.comments.push( { body: 'I do not agree because ...
199                                 author: 'doconnor',
200                                 upvotes: 0 } )
201              p.save (function (err) {...
208              })
209          }
210  } )
```

- **Add sub-document to array as normal, then save the parent document (post).**

Adding Sub-documents

- **Each sub-document is assigned its own _id value.**

```
> db.posts.find({ "_id" : ObjectId("5510117c1a9f03cd1ed38a6c") }  )
{ "__v" : 2, "_id" : ObjectId("5510117c1a9f03cd1ed38a6c"), "comments" : [      {      "bod
y" : "I agree because ....",    "author" : "doconnor",  "upvotes" : 0,  "_id" : ObjectId("55
1020f317bf07692231caed") },     {       "body" : "I do not agree because ....",         "aut
hor" : "doconnor",      "upvotes" : 0,  "_id" : ObjectId("5510250c652eae6523d95891") } ], "l
ink" : "http://www.bbc.com/news/world-asia-30896028", "title" : "India - Tiger population se
es 30% increase.", "upvotes" : 0, "username" : "jbloggs" }
>
```

- **Special method for finding a sub-document.**

```
212  Post.findById('5510117c1a9f03cd1ed38a6c', function (err,p){
213          var c = p.comments.id('551020f317bf07692231caed')
214          console.log(c)
215          process.exit()
216  } )
```

```
$ node mongoose_demo.js
{ body: 'I agree because ....',
  author: 'doconnor',
  upvotes: 0,
  _id: 551020f317bf07692231caed }
$
```

Removing Sub-documents

- **Remove the sub-document, then save the parent**

```
218  Post.findById('5510117c1a9f03cd1ed38a6c', function (err,p){
219          p.comments.id('551020f317bf07692231caed').remove()
220          p.save (function (err) {
221              if (err) {
222                  console.log(p)
223              } else {
224                  console.log('comment removed')
225              }
226              process.exit()
227          })
228  } )
```

The lodash package

The lodash npm module

(previously called underscore)

- **Excellent iteration support for arrays, strings, objects.**
  - **Lodash promotes a functional programming style**

- **Some examples**

```
var _ = require('lodash')
var customers = [ . . . Customer objects . . . ]
var  id = 11252
// Find a customer with a particular id property value
var index = _.findIndex(customers , function(customer) {
            return customer.id == id;
       });
console.log(customers[index])
```

The lodash npm module

(previously called underscore)

```
var _ = require('lodash')
var customers = [ . . . Customer objects . . . ]
// Remove customers with a balance < 10
var elements = _.remove(customers ,  function(customer) {
        return customer.balance < 10;
    });
// elements contained remove objects
```

```
// Find a customer with a particular id property value
var index = _.findIndex(customers , function(customer) {
        return customer.id == 12241;
    });
console.log(customers[index])
```