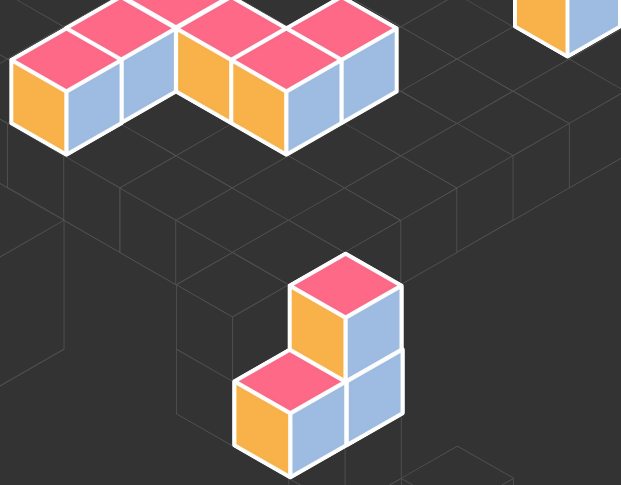


MongoDB & Node.js

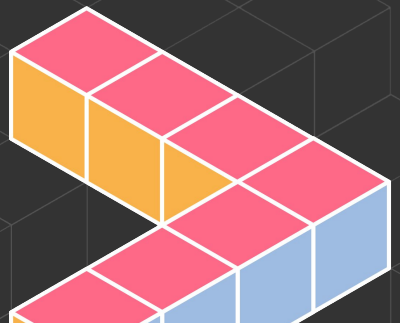
Zero to Hero in 60 Minutes

Ken W. Alger

Developer Advocate, MongoDB

@kenwalger

Our Quest



Project Overview

- MEAN Stack - without the A
- Tools
  - MongoDB
  - Node.js & NPM
  - Express
- https://github.com/kenwalger/scc2018

Sending Leafie on a Quest for food

- A document data store
- Stores JSON-like documents with schemas
- Initially released in 2009, now on version 4.0 with 4.2 coming soon

Document Model

RDBMS

PERSON

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rome

NO RELATION

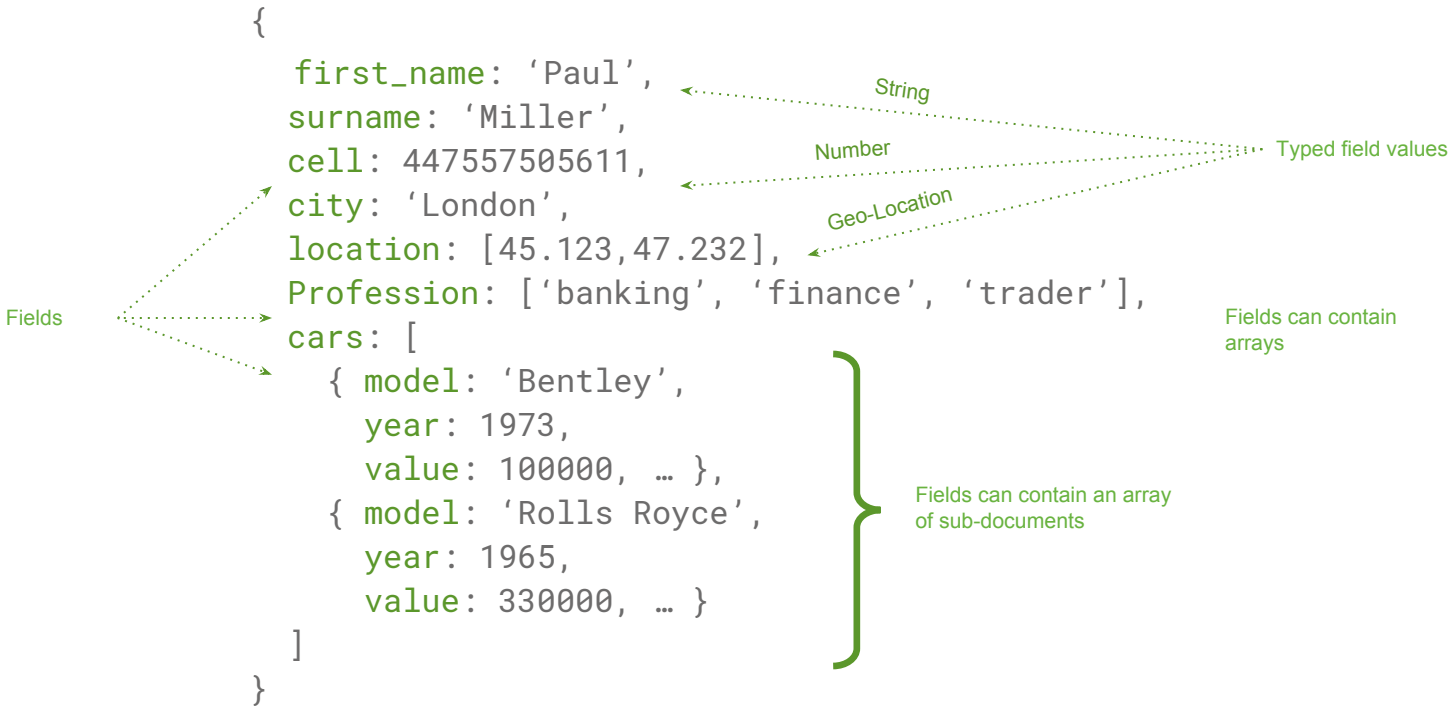
CAR

Car_ID	Model	Year	Value	Pers_ID
101	Bently	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

MongoDB

```
{
  first_name: 'Paul',
  surname: 'Miller',
  city: 'London',
  location: [45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```

DOCUMENTS ARE RICH DATA STRUCTURES



DOCUMENTS ARE FLEXIBLE

Documents in the same product catalog collection in MongoDB

Product Catalog

```
{
   product_name: 'Acme Paint',
  color: ['Red', 'Green'],
  size_oz: [8, 32],
  finish: ['satin', 'eggshell']
}
```

```
{
  product_name: 'T-shirt',
  size: ['S', 'M', 'L', 'XL'],
  color: ['Heather Gray' … ],
  material: '100% cotton',
  wash: 'cold',
  dry: 'tumble dry low'
}
```

```
{
  product_name: 'Mountain Bike',
  brake_style: 'mechanical disc',
  color: 'grey',
  frame_material: 'aluminum',
  no_speeds: 21,
  package_height: '7.5x32.9x55',
  weight_lbs: 44.05,
  suspension_type: 'dual',
  wheel_size_in: 26
}
```

- A cross-platform runtime environment for developing server-side applications with JavaScript.
- Uses an event-driven, non-blocking I/O model

- npm (the node package manager) allows for the sharing and reuse of code and software written by other developers in the JavaScript community

express

- A web framework for Node.js.
- Allows developers to build scalable, feature-rich applications, websites and more on top of Node.
- Provides built in tools and functionality to take care of common tasks, like routing (or handling requests at different URL paths), serving static files, and dynamic templating to render HTML pages.
- Simplifies the HTTP request operations.

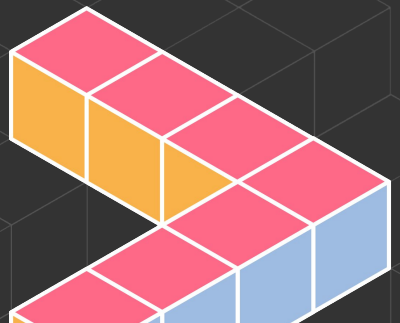
Why MEAN/MERN Stack?

- Same language (JavaScript) for all development
- Backend, API Focused

So What?

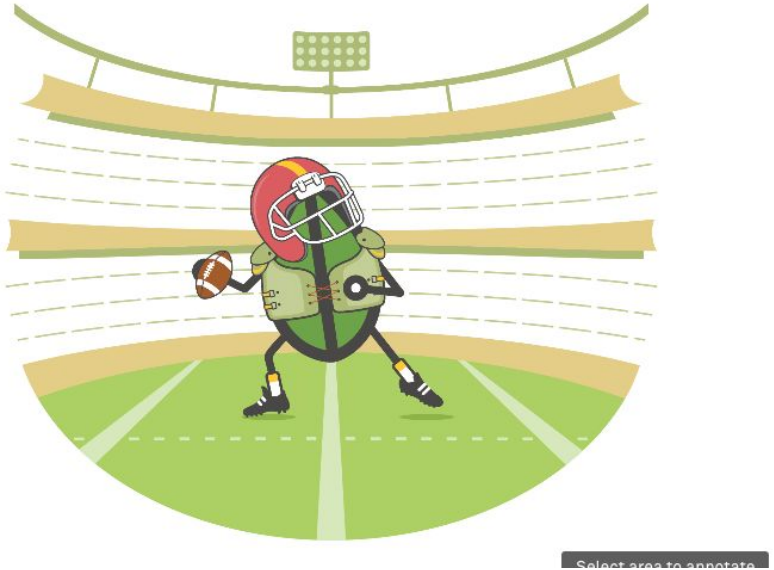
- Reduced development cost
- Fast MVP development and scalability
- Increased developer flexibility and efficiency
- Excellent performance
- Large talent pool

Let’s Get Started



Creating a Server

- package.json
- server.js



Server.js (Node.js only)

```
const http = require('http');
const port = process.env.PORT || 3000;

const requestHandler = (request, response) => {
   console.log(request.url);
   response.end('Hello Node.js Server!');
};
const server = http.createServer(requestHandler);
server.listen(port, (err) => {
   if (err) {
       return console.log('Doh! Something went wrong: ', err);
   }

   console.log(`Node.js server is running on ${port}`);
});
```

Adding Express

app.js

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;

app.use((request, response, next) => {
    response.status(200).json({
        message: "You're becoming a hero!"
    });
});

// Start the web server
app.listen(port, ()=> {
    console.log('Express.js server is listening on Port %s.', port);
});
```

Routes

Handling Requests

- HTTP Methods
  - POST (**`CREATE`**)
  - GET (**`READ`**)
  - PATCH (**`UPDATE`**)
  - DELETE (**`DELETE`**)
- Routing in our app

./api/routes/restaurant.js

```
const express = require('express');
const router = express.Router();

/* ===================================== */
// Route Definitions
/* ===================================== */

// GET routes
router.get('/', (request, response, next) => {
    console.log("You made it to the router. Nice work!");
    response.send("API index");
});

router.get('/restaurants',(request, response, next) => {
    console.log("You're on the quest for heros.");
    response.send("Restaurant index");
});

// POST routes
// PUT routes
// DELETE routes
// Export Router for use in app.js
module.exports = router;
```

Update app.js

- Add handler
  - `const restaurantRoutes = require('./api/routes/restaurant');`
- Use Route
  - `app.use('/', resaurantRoutes);`

app.js

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;
const restaurantRoutes = require('./api/routes/restaurants');

app.use('/', restaurantRoutes);

app.use((request, response, next) => {
    response.status(200).json({
        message: "You're becoming a hero!"
    });
});

// Start the web server
app.listen(port, ()=> {
    console.log('Express.js server is listening on Port %s.', port);
});
```

Schema

Data Definition

- Designing the schema

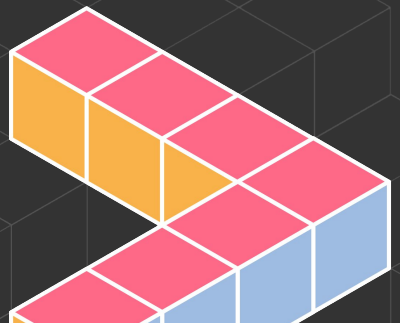
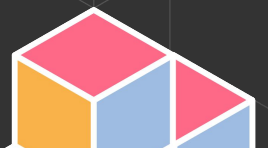
```
{
"location":
   {"coordinates":
       [
         -122.3163023,
         47.6109314
       ],
       "type":"Point"
   },
"name" : "Rhein Haus Seattle"
}
```

Could this exist together?

```
{
"location":
   {"coordinates":
       [
         -122.3163023,
         47.6109314
       ],
       "type":"Point"
   },
"name" : "Rhein Haus Seattle",
"cuisine" : "German"
}
```

Adding in MongoDB



Native JavaScript & MongoDB

- Dedicated Node.js Driver
  - The official MongoDB Node.js driver provides both callback-based and Promise-based interaction with MongoDB, allowing applications to take full advantage of the new features in ES6. The 2.x series of the driver is powered by a brand new core driver.
- MongoClient

By the way, MongoDB supports drivers for other languages as well, such as Java, Python, C/C++, PHP, and others.

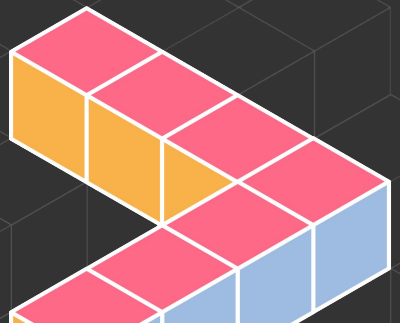
./api/data/db.js

```
let MongoClient = require('mongodb').MongoClient;
const dbName = 'dining';
const mdbPort = 27017;
const url = 'mongodb://localhost:' + mdbPort + '/' + dbName;

class Connection {
    constructor(uri, name) {
        this.db = null;
        this.uri = uri;
        this.name = name;}
    connect() {
        if(this.db) {
            return Promise.resolve(this.db);
        } else {
            return MongoClient.connect(this.uri)
                .then(client => {
                    this.db = client.db(this.name);
                    return this.db;
                });

}}}
module.exports = new Connection(url, dbName);
```

Routing Data



./api/routes/restaurants.js

```
const express = require('express');
const router = express.Router();

const db = require('../data/db').db;
let restaurants = db.collection('restaurants');

/* ======================================= */
// Route Definitions
/* ======================================= */

// GET routes
router.get('/', (request, response, next) => {
    console.log("You made it to the router. Nice work!");
    response.send("API index");
});

router.get('/restaurants',(request, response, next) => {
    console.log("You're on the quest for heros.");
       restaurants.find().toArray(function(err, results) {
        response.send(results);
    });
});
```

./api/routes/delis.js

```
// POST Routes
router.post('/restaurants/', (request, response) => {
    restaurants.insertOne(request.body, (err, result) => {
        if (err) return console.log(err);
        response.send("Here's the data that was saved: " + JSON.stringify(request.body));
    });
});

// PUT routes

// DELETE routes

// Export Router for use in app.js
module.exports = router;
```

Updating app.js… again

```
// MongoDB & Node.js Sample API

/* ======================================= */
// Express Configuration
/* ======================================= */
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;
const bodyParser = require('body-parser');

// support parsing of application/x-www-form-urlencoded post data
app.use(bodyParser.urlencoded({extended: true}));

// support parsing of application/json type post data
app.use(bodyParser.json());

const db = require('./api/data/db');
```

Note: in Express version 4, body-parser is included in the Express package. *But* it was there in v2.x as well, but not in 3.x, so... yeah. You'll see it used as `bodyParser.urlencoded` and `express.urlencoded` depending on coding style.

And adding in the database...

```
const db = require('./api/data/db');
const data = require('./api/data/restaurants');

db.connect().then(function(db){
            const restaurantRoutes = require('./api/routes/restaurants');

            app.use('/', restaurantRoutes);

            app.use((request, response, next) => {
                response.status(200).json({
                    message: "You're becoming a hero!"
                });
            });

            // Start the web server
            app.listen(port, () => {
                console.log('Express.js server is listening on Port %s.', port);
            });

});
```

Testing with Postman

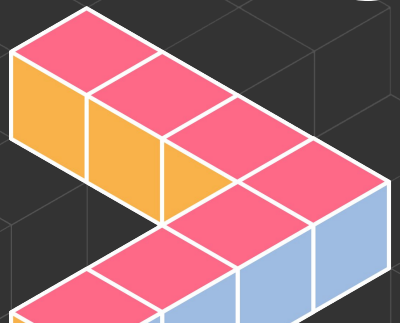
Add a new document:

```
{
    "location":
        {"coordinates":
            [
               -122.3165271,
               47.6170893
            ],
         "type":"Point"
        },
    "name":"Scratch Deli"
}
```

Looking at the data

Congratulations!!!

That’s the Magic of the MEAN/MERN Stack



Next Steps:

- MongoDB University Courses (university.mongodb.com)
  - M001: MongoDB Basics
  - M101JS: MongoDB for Node.js Developers

Final project code available at:
https://github.com/kenwalger/scc2018

Thank You!