

The background features a dark gray isometric grid of cubes. Some cubes are highlighted with colors: a large cluster of pink and blue cubes in the upper right, and a smaller cluster of pink and blue cubes in the lower right. The text is positioned on the left side of the image.

MongoDB & Node.js

Zero to Hero in 60 Minutes



Ken W. Alger

Developer Advocate, MongoDB

@kenwalger

The background is a dark gray grid of isometric cubes. Several individual cubes are scattered across the grid, each with a different color: orange, pink, and light blue. Some cubes are stacked on top of each other, creating a sense of depth. The text "Our Quest" is centered in the middle of the image in a white, sans-serif font.

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

CAR

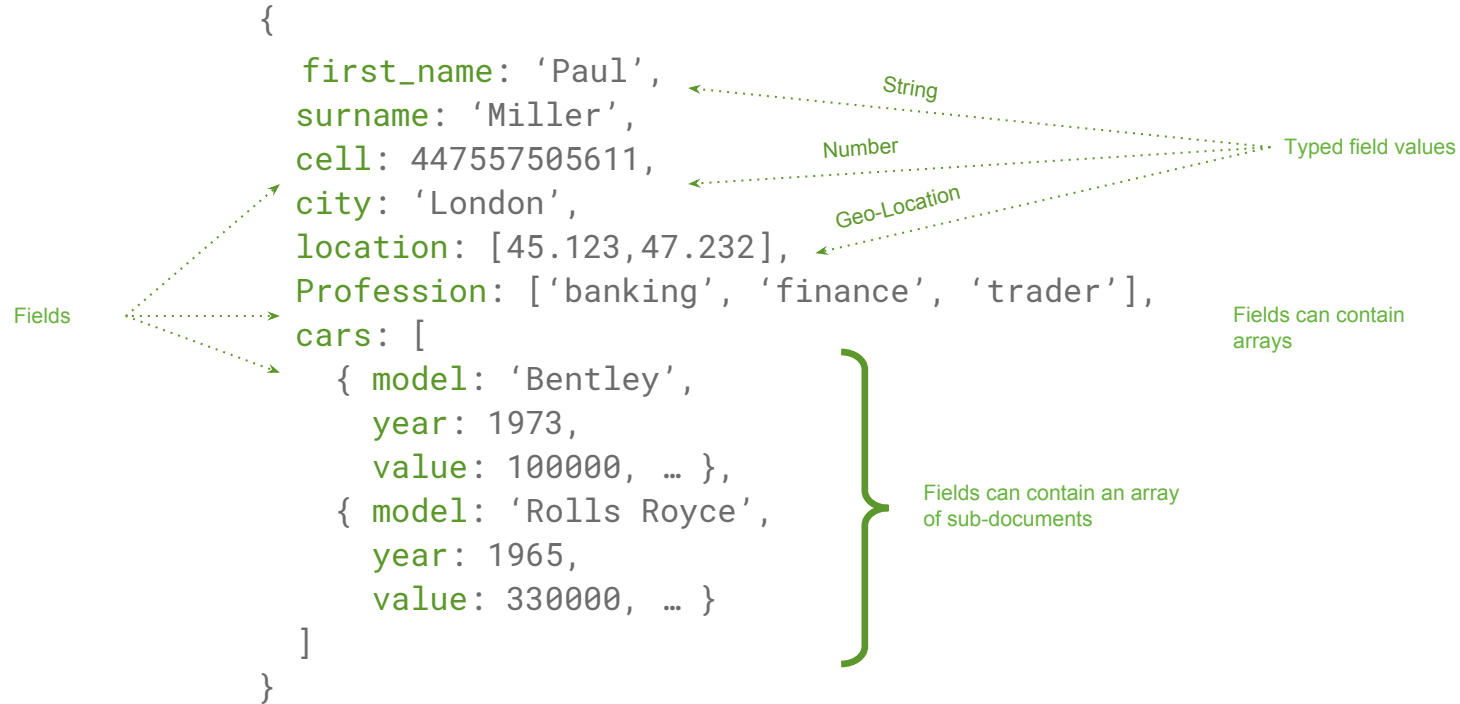
Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

NO RELATION

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

The background is a dark gray grid of isometric cubes. Scattered throughout are several Tetris-like pieces: a 2x2 square piece in the top left, a single cube in the top right, a single cube on the middle left, a 3x2 L-shaped piece in the bottom left, and a 2x2 square piece in the bottom right. All pieces have orange, pink, and blue faces.

Let's Get Started

Creating a Server

- package.json
- server.js



Select area to annotate

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}`);
});
```

The background is a dark gray grid of isometric cubes. Several cubes are highlighted with colors: orange, pink, and blue. Some cubes are stacked to form larger structures, such as a 2x2x2 cube in the top left and a 2x2x2 cube in the bottom right. A large, irregular shape made of colored cubes is in the bottom left. The text "Adding Express" is centered in the middle of the image.

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

The background is a dark gray isometric grid of cubes. Scattered throughout are several 3D blocks. Most are composed of three visible faces: a pink top face, an orange side face, and a light blue side face. One block in the lower-left is composed of five such faces, forming a more complex shape. The word 'Routes' is centered in a white, sans-serif font.

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('/', restaurantRoutes);`

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

The background is a dark gray field filled with a grid of isometric cubes. Most of these cubes are faint and gray, creating a sense of depth. Scattered throughout the scene are several cubes with bright colors: pink, orange, and blue. Some of these colored cubes are isolated, while others are grouped together to form small, simple structures. For example, in the bottom left, there is a small cluster of cubes forming an L-shape, with pink and orange faces visible. Other smaller groups of colored cubes are located in the top left, top right, middle left, and bottom right areas.

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"
}
```

The background is a dark gray grid of isometric cubes. Several cubes are highlighted with colors: orange, pink, and blue. In the top left, there is a small cluster of three cubes (two orange, one pink). In the top right, there is a single orange cube. On the left side, there is a single pink cube. In the bottom left, there is a larger cluster of seven cubes arranged in a 2x4 grid with the bottom-right corner missing. In the bottom right, there is a small cluster of three cubes (two orange, one pink).

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

The background is a dark gray grid of isometric cubes. Several cubes are highlighted with colors: orange, pink, and blue. In the top left, there is a small cluster of three cubes (two orange, one pink). In the top right, there is a single orange cube. On the left side, there is a single pink cube. In the bottom left, there is a larger cluster of seven cubes arranged in a 2x4 grid with the bottom-right corner missing (three orange, four pink). In the bottom right, there is a small cluster of three cubes (two orange, one pink).

./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



The background is a dense field of green isometric cubes. Scattered throughout are several smaller cubes with different colors: orange, pink, and blue. Some cubes are stacked, while others are single. The text 'Congratulations!!!' is centered in the middle of the image.

Congratulations!!!

The background is a dark gray field with a subtle, repeating pattern of light gray isometric cubes and hexagons. Scattered throughout are several 3D cubes in isometric perspective. Some are solid colors (orange, blue, pink), while others are combinations of these colors on their visible faces. One notable cluster of cubes is in the bottom left, forming a larger, more complex geometric shape.

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>

The background is a dark gray grid of isometric cubes. Several cubes are highlighted with colors: orange, pink, and blue. In the top left, there is a small cluster of three cubes (two orange, one pink). In the top right, there is a single orange cube with a pink top face. On the left side, there is a single pink cube with a blue right face. In the bottom left, there is a larger cluster of seven cubes arranged in an 'L' shape (four orange, three pink, and one blue). In the bottom right, there is a small cluster of three cubes (one orange, one pink, one blue).

Thank You!