

# Web Application Development

---

Produced  
by

David Drohan ([ddrohan@wit.ie](mailto:ddrohan@wit.ie))

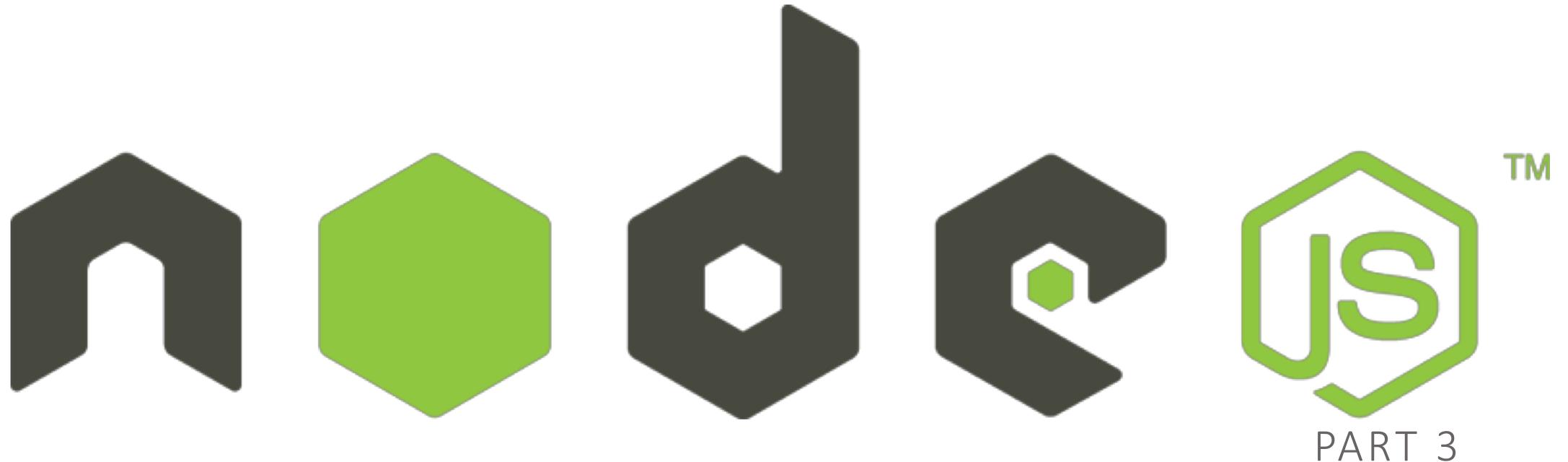
Department of Computing & Mathematics  
Waterford Institute of Technology

<http://www.wit.ie>



Waterford Institute *of* Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE





---

mongoose & mongoDB

# Table of Contents

---

1. Databases Overview
2. MongoDB Overview
  - Installation
  - Structure and documents
  - Hosting locally MongoDB
  - Console CLI
  - RoboMongo, MongoVUE, Umongo
  - Executing queries on MongoDB data
3. Mongoose Overview
  - Mongoose Models
    - Types of properties
  - Mongoose CRUD operations
    - save, remove, find
  - Mongoose Queries



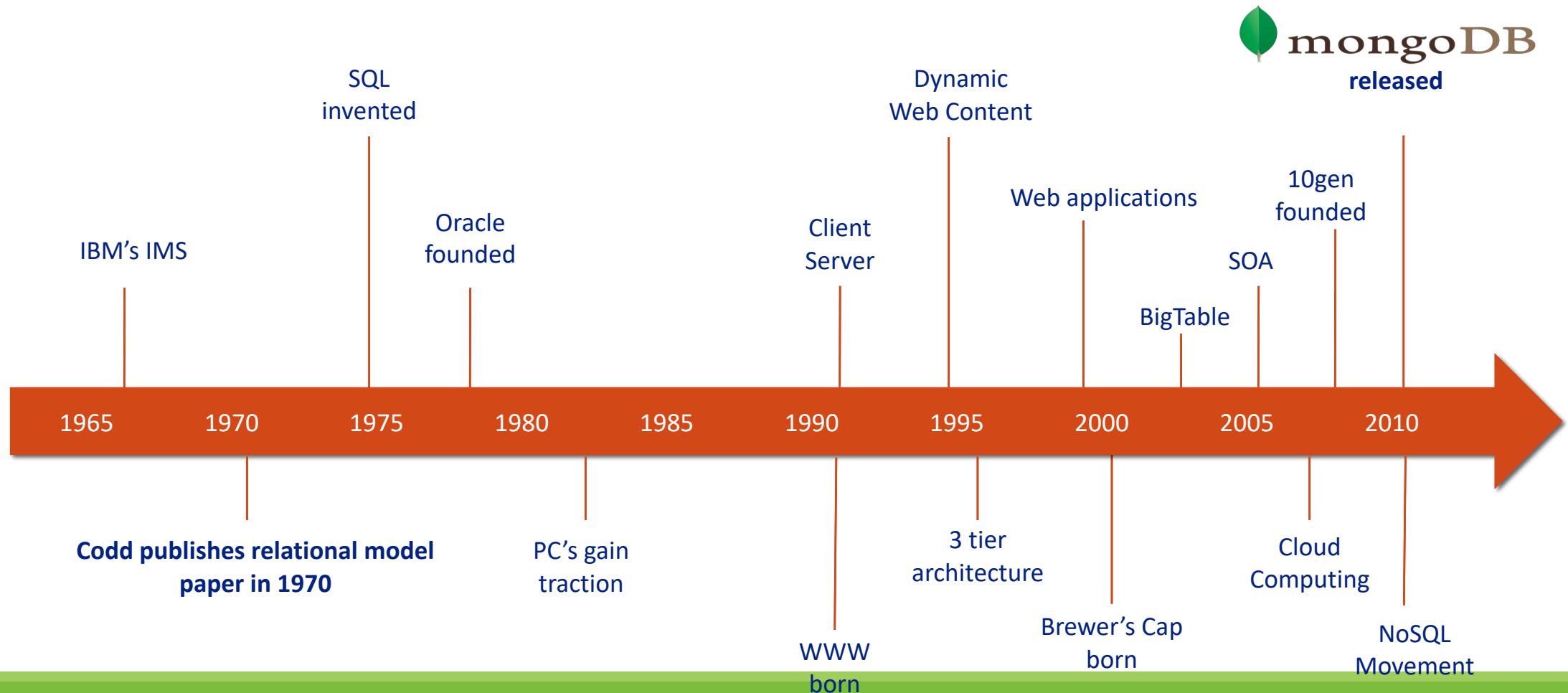
# Databases

---

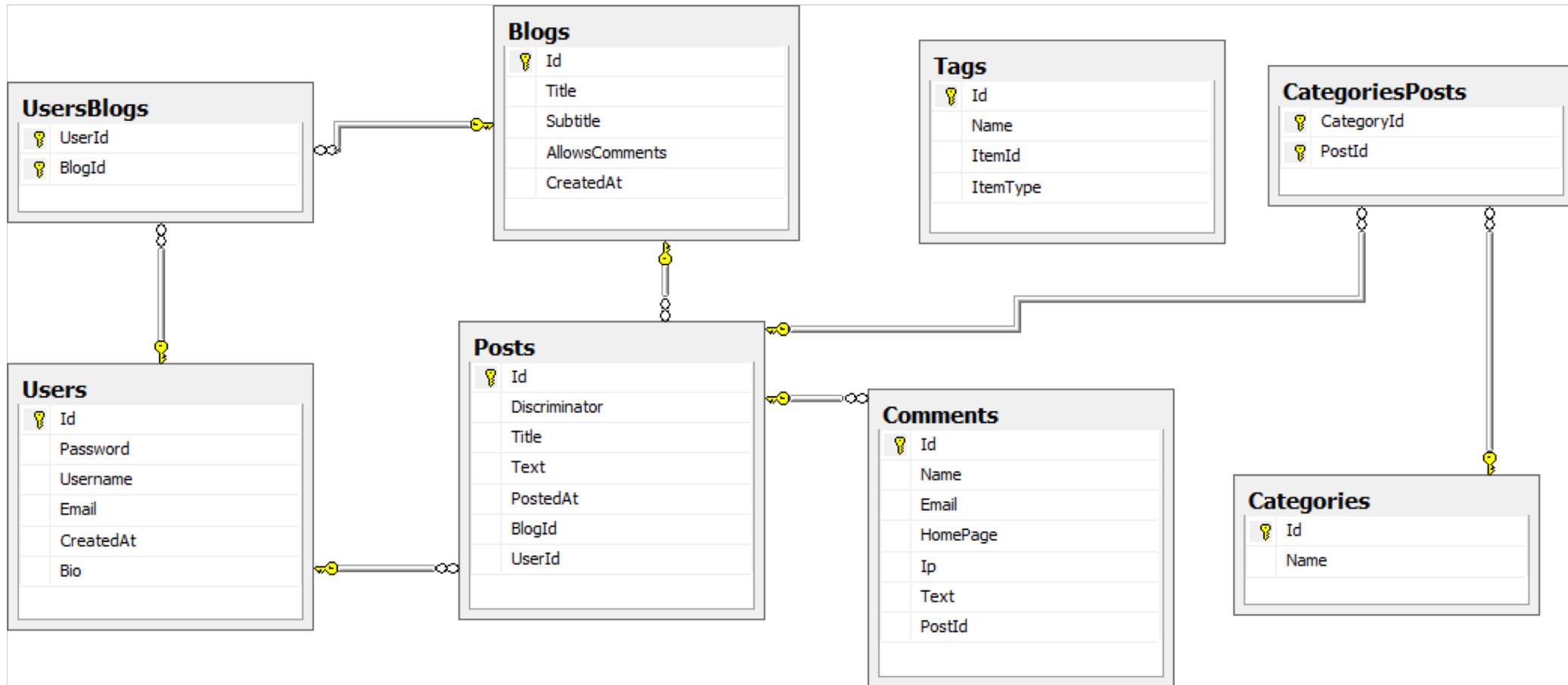
HISTORY & BACKGROUND



# Dawn of the Databases



# Relational Databases



# Relational Databases Strengths

---

Data stored in a RDBMS is very compact (disk was more expensive)

SQL and RDBMS made queries flexible with rigid schemas

Rigid schemas helps optimize joins and storage

Massive ecosystem of tools, libraries and integrations

Been around 40 years!

# Enter Big Data

---

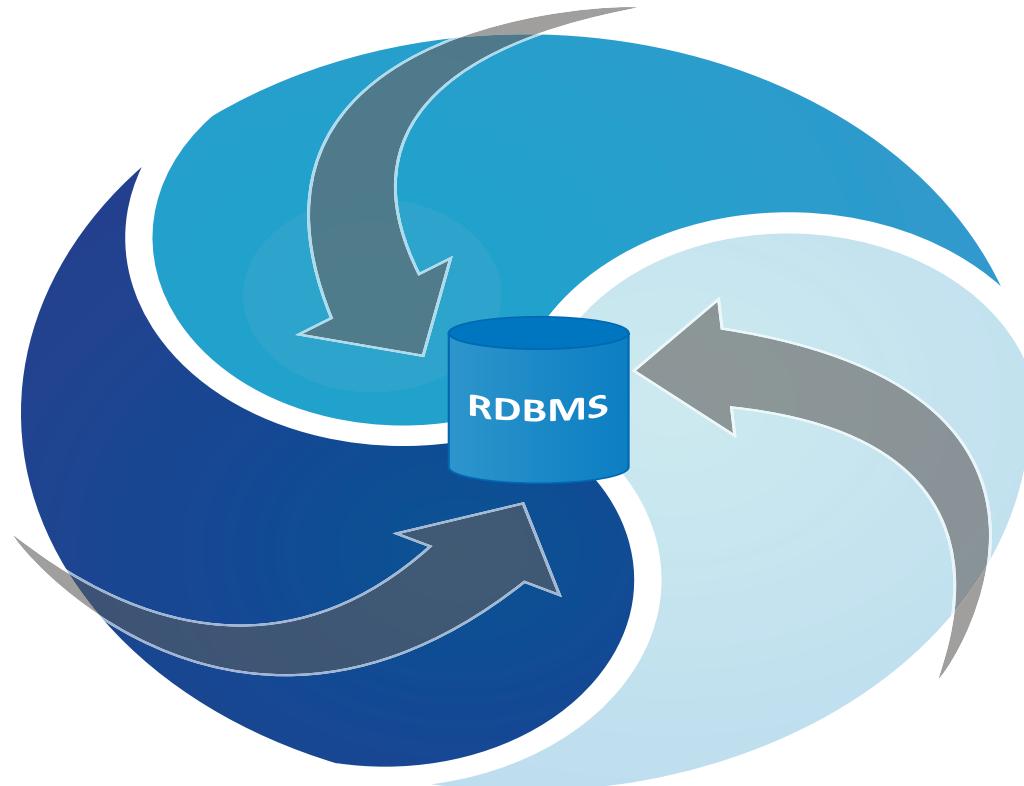
Gartner uses the 3Vs to define

- Volume - Big/difficult/extreme volume is relative
- Variety
  - Changing or evolving data
  - Uncontrolled formats
  - Does not easily adhere to a single schema
  - Unknown at design time
- Velocity
  - High or volatile inbound data
  - High query and read operations
  - Low latency

# RDBMS Challenges

## DATA VARIETY & VOLATILITY

- Extremely difficult to find a single fixed schema
- Don't know data schema a-priori



## VOLUME & NEW ARCHITECTURES

- Systems scaling horizontally, not vertically
- Commodity servers
- Cloud Computing

## TRANSACTIONAL MODEL

- N x Inserts or updates
- Distributed transactions

# Enter NoSQL

---

Modern NoSQL theory and offerings started in early 2000s  
(idea around much longer)

Modern usage of term introduced in 2009

NoSQL = Not Only SQL

A collection of very different products

Alternatives to relational databases when they are a bad fit

Motives

- Horizontally scalable
- Flexibility

# NoSQL Databases – Key-Value

---

## Key-Value Stores

- Key-Value Stores
- Value (Data) mapped to a key (think primary)

## Big Table Descendants

- Looks like a distributed multi-dimension map
- Data stored in a column oriented fashion
- Predominantly hash based indexing

## Document oriented stores

- Data stored as either JSON or XML documents

# 10gen & MongoDB

---

2007

- Eliot Horowitz & Dwight Merriman tired of reinventing the wheel
- 10gen founded
- MongoDB Development begins

2009

- Initial release of MongoDB

73M+ in funding

- Funded by Sequoia, NEA (New Enterprise Associates), Union Square Ventures, Flybridge Capital

# MongoDB

---

OBJECT-DOCUMENT MODEL



{ name: mongo, type: DB }



# Overview: MongoDB – What is it?

---

MongoDB is an open-source **document store** database

- The leading NoSQL database (see next slide)
- Save JSON-style objects with dynamic schemas

Support many features:

- Support for indices
- Rich, document-based queries (CRUD operations)
- Flexible aggregation and data processing
- Store files of any size without complicating your stack
- Fast in-place updates

# DB-Engines Ranking (October 2018)

346 systems in ranking, October 2018

Rank			DBMS	Database Model	Score		
Oct 2018	Sep 2018	Oct 2017			Oct 2018	Sep 2018	Oct 2017
1.	1.	1.	Oracle 	Relational DBMS	1319.27	+10.15	-29.54
2.	2.	2.	MySQL 	Relational DBMS	1178.12	-2.36	-120.71
3.	3.	3.	Microsoft SQL Server 	Relational DBMS	1058.33	+7.05	-151.99
4.	4.	4.	PostgreSQL 	Relational DBMS	419.39	+12.97	+46.12
5.	5.	5.	MongoDB 	Document store	363.19	+4.39	+33.79
6.	6.	6.	DB2 	Relational DBMS	179.69	-1.38	-14.90
7.	↑ 8.	↑ 9.	Redis 	Key-value store	145.29	+4.35	+23.24
8.	↓ 7.	↑ 10.	Elasticsearch 	Search engine	142.33	-0.28	+22.09
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	136.80	+3.41	+7.35
10.	10.	↓ 8.	Cassandra 	Wide column store	123.39	+3.83	-1.40
11.	11.	11.	SQLite 	Relational DBMS	116.74	+1.29	+4.76

# Overview: MongoDB Key Features

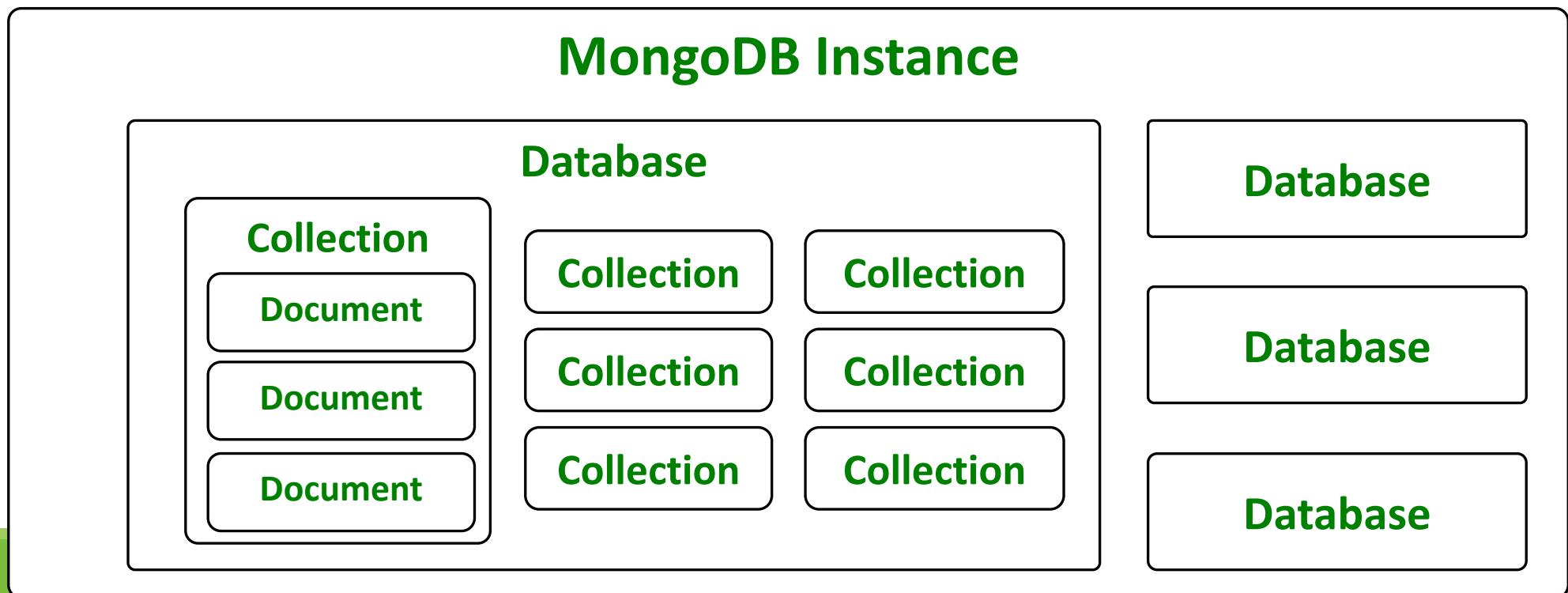
---

- Data stored as documents (JSON)
- Schema free
- CRUD operations – (Create Read Update Delete)
- Atomic document operations
- Ad hoc Queries like SQL
  - Equality
  - Regular expression searches
  - Ranges
  - Geospatial
- Secondary indexes
- Sharding (sometimes called partitioning) for scalability
- Replication – HA (High Availability) and read scalability

# Overview: MongoDB Data Model

A MongoDB instance can have many databases

- A database can have many collections
  - A collection can have many documents



# Overview: MongoDB Documents

Documents in MongoDB are JSON objects

**Module Name:** Databases

**Students:** 250

**Titles:**

- Intro to SQL
- NoSQL databases
- Entity Framework

**Evaluation:**

- Homework: 10
- Exam: 60
- Teamwork: 20
- Attendance: 10



```
{  
  "module_name": "Databases",  
  "students": 250,  
  "titles": [  
    "Intro to SQL",  
    "NoSQL databases",  
    "DB Performance"  
  ],  
  "evaluation": {  
    "homework": 10,  
    "exam": 60,  
    "teamwork": 20,  
    "attendance": 10  
  }  
}
```

# Overview: MongoDB Documents

---

MongoDB documents can have many types of values

- Numbers, Booleans
- Strings, Object Ids
- Dates
- Objects
  - Also known as nested documents
- Arrays

**17, 3.14, true/false**

**"David Drohan"**

**2014-09-01T14:58:48.126Z**

```
{  
  "username": "daveydz",  
  "accessLevel": 2  
}
```

**[ 'bananas', 'oranges' ]**



{ name: mongo, type: DB }

# MongoDB Installation

---

INSTALLING & USING MONGODB



# MongoDB Installation

---

Download MongoDB from the official web site:

- <https://www.mongodb.org/downloads>
- Installers for all major platforms

How to start MongoDB?

- Go to installation folder and run **mongod**
- Paths need some configuration **\$ cd C:\MongoDb\bin**
  - You should specify databases path
  - or use default path and create folder **C:\data\db**

```
$ mongod --dbpath D:\mongodb\data
```

```
$ mongod
```

# How to use MongoDB

---

Connect to a `mongodb`

```
$ mongo
```

Select a database

```
$ use <database name>
```

Show database name

```
$ db
```

Show all databases

```
$ show dbs
```

# Authentication

How to create users in a database?

```
$ use ssd4db
$ db.createUser({
  "user": "daveyd",
  "pwd": "ssd412345",
  "roles": ["readWrite", "dbAdmin"]
})
```



Database user roles:

- **read**, **readWrite**, **dbAdmin**, **dbOwner**, **userAdmin**
- **userAdminAnyDatabase**

Login: **db.auth("daveyd", "ssd412345")**



{ name: mongo, type: DB }

# MongoDB Queries

---

INSTALLING & USING MONGODB



# MongoDB Queries – Read

Read query

```
$ db.courses.find({ level: { $gt: 2 }}, { module_name: true }).limit(5);
```

collection

filters

projection

cursor modifier

Same as (in SQL):

```
SELECT module_name
FROM courses
WHERE level > 2
LIMIT 5
```

# MongoDB Queries – Read (2)

```
$ db.courses.find({  
  module_name: "Databases"  
});
```

```
$ db.courses.find({  
  $or: [{ level: { $gt: 1 } }, { level: { $lt: 3 } }]  
});
```

## Operators:

- Comparison – **\$gt**, **\$gte**, **\$et**, **\$lt**, **\$lte**
- Logical – **\$and**, **\$nor**, **\$not**, **\$or**
- Element – **\$exists**, **\$type**

# MongoDB Queries – Insert

Insert query

```
$ db.courses.insert({  
  module_name: "Databases",  
  description: "Databases description",  
  level: 3  
})
```

Same as (in SQL):

```
INSERT INTO courses (module_name, description, level)  
VALUES ("Databases", "Databases description", 3)
```

# MongoDB Queries – Bulk Insert

Inserting bulk data in MongoDB

```
var bulk = db.courses.initializeUnorderedBulkOp();
bulk.insert( { name: "ASP.NET MVC", level: 3 } );
bulk.insert( { item: "Web Development", level: 3 } );
bulk.insert( { item: "SPA Applications", level: 2 } );
bulk.execute();
```

# MongoDB Queries – Update

Update query

```
$ db.courses.update(  
  { module_name: { $set: "Databases" }},  
  { $set: { description: "Concepts, MSSQL, MySQL, Oracle"}},  
  { multi: true }  
)
```

Same as (in SQL):

```
UPDATE courses  
SET description = "Concepts, MSSQL, MySQL, Oracle"  
WHERE module_name = "Databases"
```

# MongoDB Queries – Delete

---

Delete query

```
$ db.courses.remove({ module_name: "Databases" });
```

Same as (in SQL):

```
DELETE FROM courses  
WHERE module_name = "Databases"
```

# MongoDB Queries – Aggregation

Aggregation is operation that process data records and return computed results

```
$ db.courses.aggregate([
  { $match: { isActive: true } },
  { $group: { level: "$level", students: { $sum: "$studentsCount" } } }
])
```

Result:

```
[
  { level: 1, students: 460 },
  { level: 2, students: 350 },
  { level: 3, students: 200 }
]
```

# MongoDB Management Tools

---

MongoDB is an open-source DB system

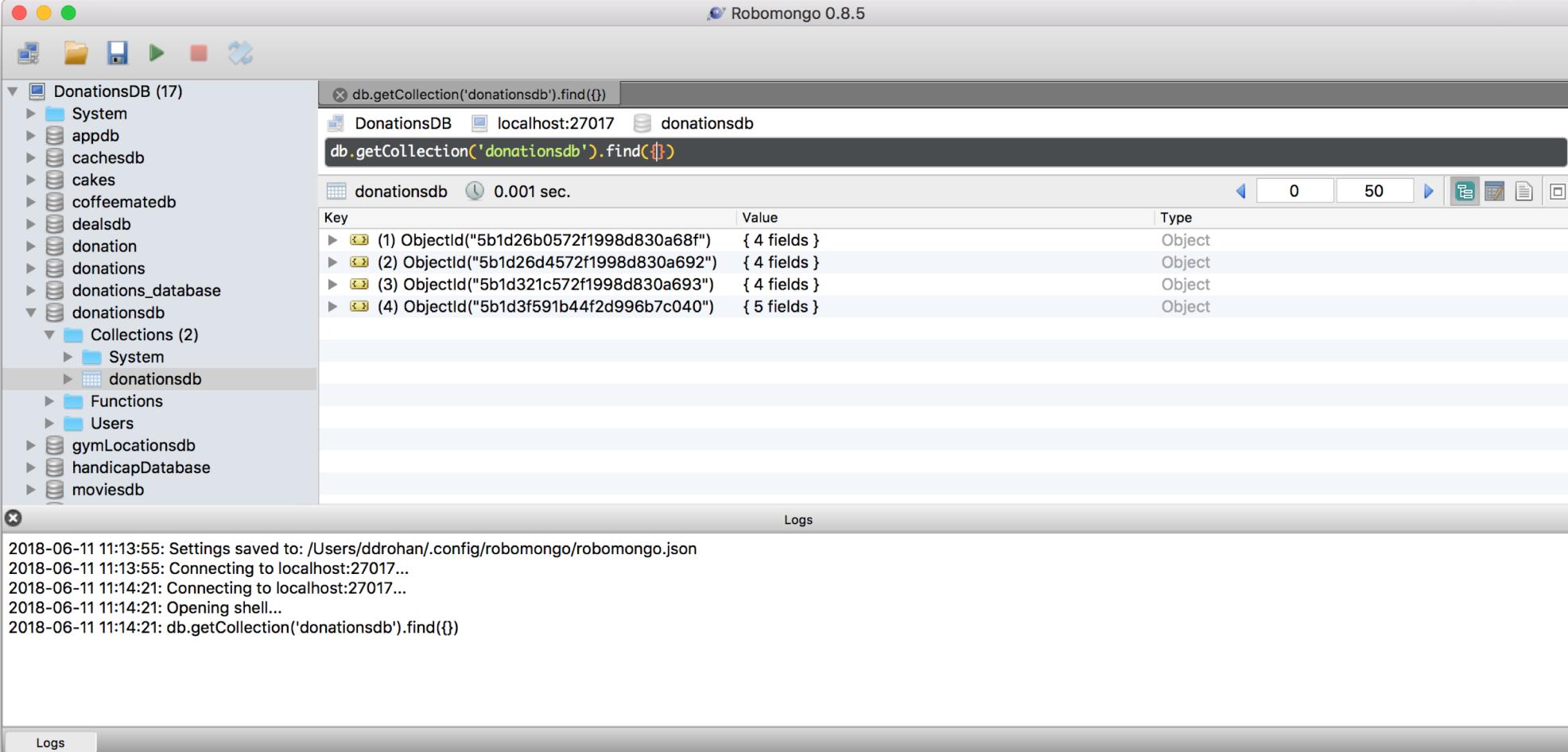
- So there are many available viewers

Some, but not all are:

- **MongoDB CLI**
  - Comes with installation of MongoDB
  - Execute queries inside the CMD/Terminal
- **MongoVUE & UMongo & Robomongo**
  - Provide UI to view, edit and remove DB documents
  - Execute queries inside the tool



# MongoDB - Robomongo



The screenshot shows the Robomongo application interface. On the left is a sidebar with a tree view of databases and collections. The 'DonationsDB' database is selected, showing its contents. In the main panel, a query builder window is open with the command `db.getCollection('donationsdb').find({})`. Below the query, the results are displayed in a table:

Key	Value	Type
► (1) ObjectId("5b1d26b0572f1998d830a68f")	{ 4 fields }	Object
► (2) ObjectId("5b1d26d4572f1998d830a692")	{ 4 fields }	Object
► (3) ObjectId("5b1d321c572f1998d830a693")	{ 4 fields }	Object
► (4) ObjectId("5b1d3f591b44f2d996b7c040")	{ 5 fields }	Object

At the bottom of the interface, there is a 'Logs' pane displaying the following log entries:

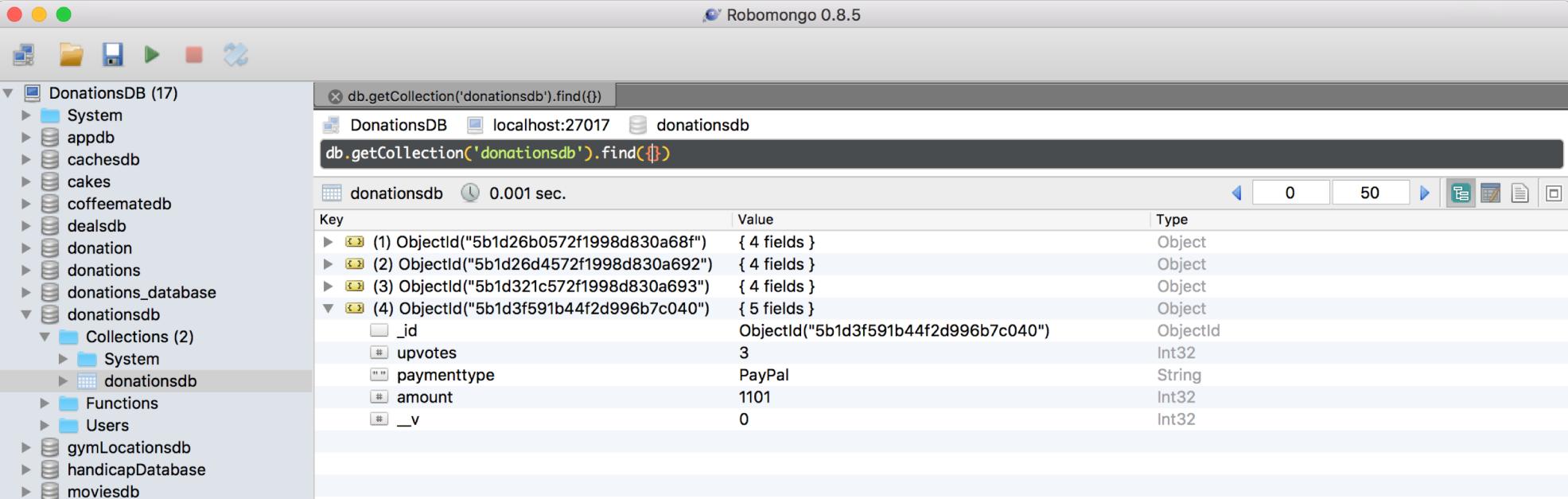
```

2018-06-11 11:13:55: Settings saved to: /Users/ddrohan/.config/robomongo/robomongo.json
2018-06-11 11:13:55: Connecting to localhost:27017...
2018-06-11 11:14:21: Connecting to localhost:27017...
2018-06-11 11:14:21: Opening shell...
2018-06-11 11:14:21: db.getCollection('donationsdb').find({})

```

# MongoDB - Robomongo

Robomongo 0.8.5



Key	Value	Type
1 ObjectId("5b1d26b0572f1998d830a68f")	{ 4 fields }	Object
2 ObjectId("5b1d26d4572f1998d830a692")	{ 4 fields }	Object
3 ObjectId("5b1d321c572f1998d830a693")	{ 4 fields }	Object
4 ObjectId("5b1d3f591b44f2d996b7c040")	{ 5 fields }	Object
_id	ObjectId("5b1d3f591b44f2d996b7c040")	ObjectId
upvotes	3	Int32
paymenttype	PayPal	String
amount	1101	Int32
__v	0	Int32

Logs

```

2018-06-11 11:13:55: Settings saved to: /Users/ddrohan/.config/robomongo/robomongo.json
2018-06-11 11:13:55: Connecting to localhost:27017...
2018-06-11 11:14:21: Connecting to localhost:27017...
2018-06-11 11:14:21: Opening shell...
2018-06-11 11:14:21: db.getCollection('donationsdb').find({})

```

Logs

# Donationweb

---

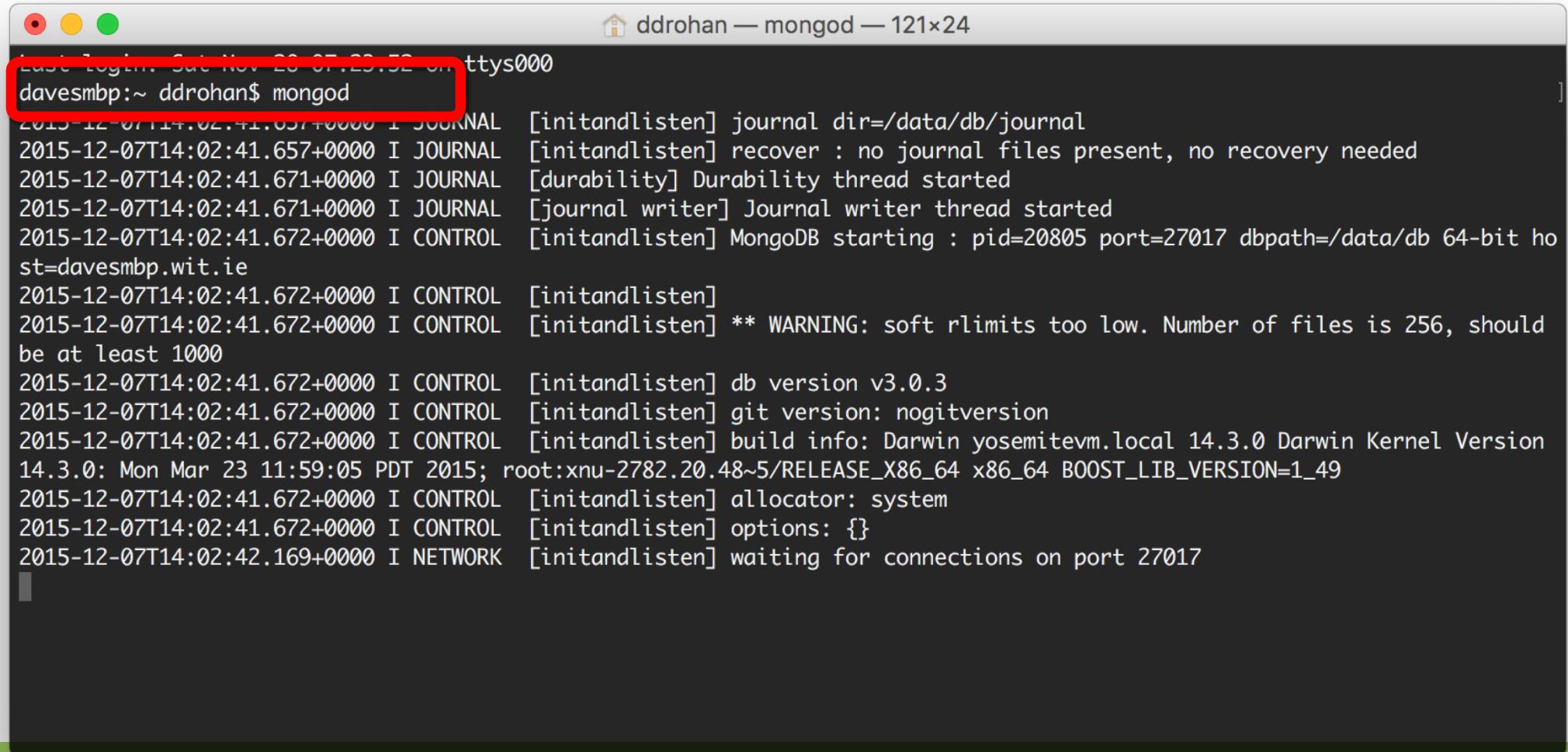
CASE STUDY EXAMPLE



{ name: mongo, type: DB }



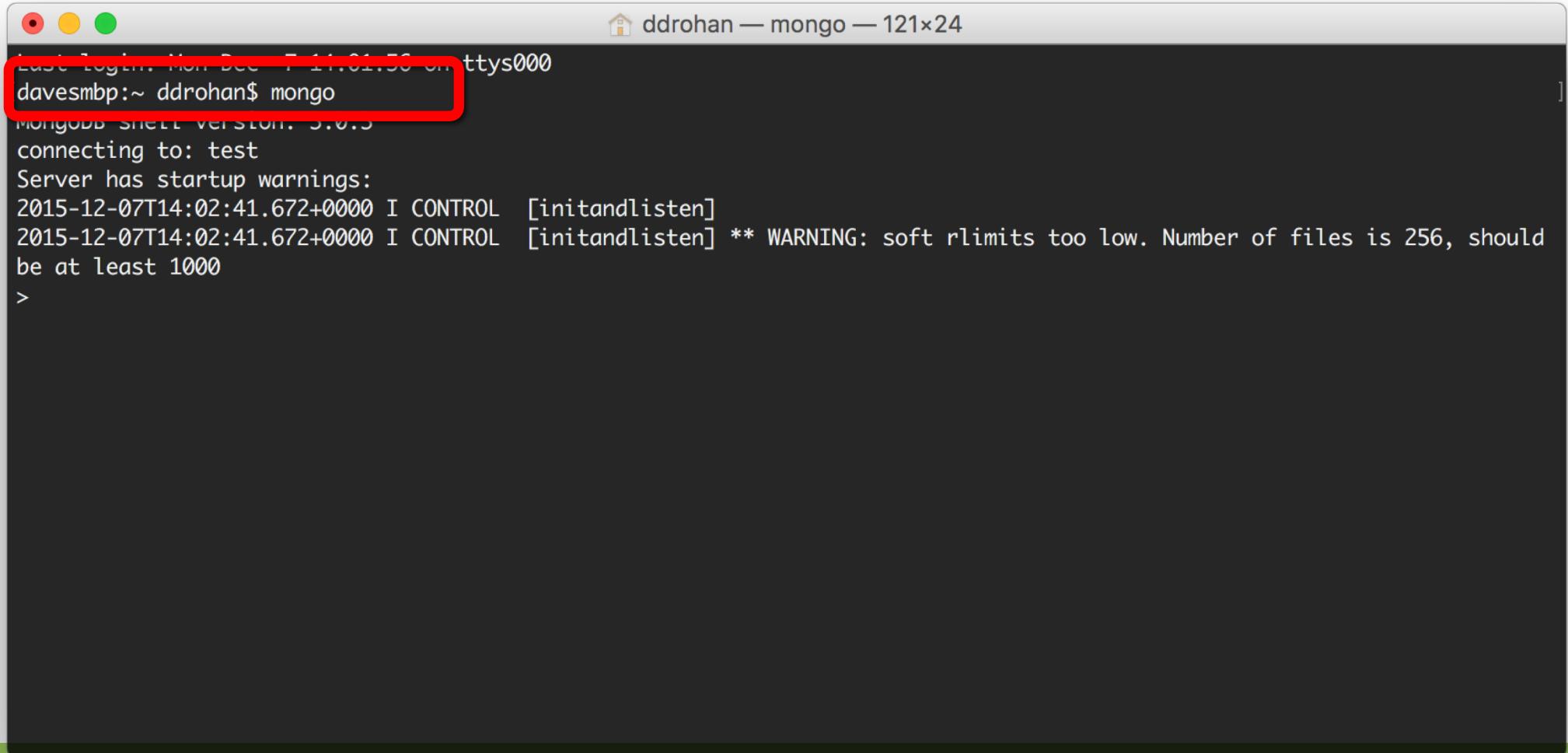
# Launch the MongoDB Server



The screenshot shows a terminal window titled "ddrohan — mongod — 121x24". The command "mongod" is being run from the user's home directory. The output log shows the server starting up, including initialization of the journal, recovery, durability, and journal writer threads. It also shows the MongoDB version, git commit, build info, and kernel details. A warning message indicates that soft rlimits are too low for file descriptors. The server finally starts listening on port 27017.

```
Last login: Sat Nov 28 07:29:52 on ttys000
davesmbp:~ ddrohan$ mongod
2015-12-07T14:02:41.657+0000 I JOURNAL [initandlisten] journal dir=/data/db/journal
2015-12-07T14:02:41.657+0000 I JOURNAL [initandlisten] recover : no journal files present, no recovery needed
2015-12-07T14:02:41.671+0000 I JOURNAL [durability] Durability thread started
2015-12-07T14:02:41.671+0000 I JOURNAL [journal writer] Journal writer thread started
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] MongoDB starting : pid=20805 port=27017 dbpath=/data/db 64-bit ho
st=davesmbp.wit.ie
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten]
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should
be at least 1000
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] db version v3.0.3
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] git version: nogitversion
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] build info: Darwin yosemitenvm.local 14.3.0 Darwin Kernel Version
14.3.0: Mon Mar 23 11:59:05 PDT 2015; root:xnu-2782.20.48~5/RELEASE_X86_64 x86_64 BOOST_LIB_VERSION=1_49
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] allocator: system
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] options: {}
2015-12-07T14:02:42.169+0000 I NETWORK [initandlisten] waiting for connections on port 27017
```

# Launch the MongoDB Client



A screenshot of a macOS terminal window titled "ddrohan — mongo — 121x24". The window shows the MongoDB shell version 3.0.5 starting up. A red box highlights the command "davesmbp:~ ddrohan\$ mongo". The output includes startup warnings about soft rlimits being too low.

```
Last login: Mon Dec 7 11:01:50 on ttys000
davesmbp:~ ddrohan$ mongo
MongoDB shell version: 3.0.5
connecting to: test
Server has startup warnings:
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten]
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
>
```

# Create/Switch to chosen Database

```
Last login: Mon Dec  7 14:01:56 on ttys000
[davesmbp:~ ddrohan$ mongo
MongoDB shell version: 3.0.3
connecting to: test
Server has startup warnings:
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten]
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
[> use donationsdb
switched to db donationsdb
[> db
donationsdb
[> show dbs
coffeematedb  0.078GB
donationsdb   0.078GB
local         0.078GB
>
```

# Search / Insert Records

```
Last login: Mon Dec  7 14:01:56 on ttys000
[davesmbp:~ ddrohan$ mongo
MongoDB shell version: 3.0.3
connecting to: test
Server has startup warnings:
2015-12-07T14:02:41.672+0000 I CONTROL  [initandlisten]
2015-12-07T14:02:41.672+0000 I CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should
be at least 1000
[> use donationsdb
switched to db donationsdb
[> db
donationsdb
[> show dbs
coffeematedb  0.078GB
donationsdb   0.078GB
local          0.078GB
[> db.donationsdb.find();
[> db.donationsdb.insert({"paymenttype":"PayPal", "amount":1000, "upvotes":0})
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.insert({"paymenttype":"Direct", "amount":1500, "upvotes":0})
WriteResult({ "nInserted" : 1 })
[>
```

# Find All Records

```
MongoDB shell version: 3.0.3
connecting to: test
Server has startup warnings:
2015-12-07T14:02:41.672+0000 I CONTROL  [initandlisten]
2015-12-07T14:02:41.672+0000 I CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should
be at least 1000
[> use donationsdb
switched to db donationsdb
[> db
donationsdb
[> show dbs
coffeeematedb 0.078GB
donationsdb 0.078GB
local 0.078GB
[> db.donationsdb.find();
[> db.donationsdb.find()
[> db.donationsdb.insert({"paymenttype":"PayPal","amount":1000,"upvotes":0})
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.insert({"paymenttype":"Direct","amount":1500,"upvotes":0})
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
>
```

# Find Records (with criteria)

```

ddrohan — mongo — 121x24
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten]
2015-12-07T14:02:41.672+0000 I CONTROL [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should
be at least 1000
[> use donationsdb
switched to db donationsdb
[> db
donationsdb
[> show dbs
coffeematedb 0.078GB
donationsdb 0.078GB
local 0.078GB
[> db.donationsdb.find();
[> db.donationsdb.insert({ "paymenttype": "PayPal", "amount": 1000, "upvotes": 0 })
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.insert({ "paymenttype": "Direct", "amount": 1500, "upvotes": 0 })
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593af87282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.find({ "_id" : "566593af87282d5d60eedc21" })
[> db.donationsdb.find({ "_id" : ObjectId("566593af87282d5d60eedc21") })
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
>

```

# Insert / Find Records (with criteria)

```

[> db
donationsdb
[> show dbs
coffeeematedb 0.078GB
donationsdb 0.078GB
local 0.078GB
[> db.donationsdb.find();
[> db.donationsdb.insert({"paymenttype":"PayPal","amount":1000,"upvotes":0})
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.insert({"paymenttype":"Direct","amount":1500,"upvotes":0})
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.find({_id" : "566593af87282d5d60eedc21"})
[> db.donationsdb.find({_id" : ObjectId("566593af87282d5d60eedc21")})
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
[> db.donationsdb.insert({"paymenttype":"PayPal","amount":1500,"upvotes":0})]
[> db.donationsdb.find({"amount":1500})
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
>
  
```

# Find all Records

```

donationsdb 0.078GB
local        0.078GB
[> db.donationsdb.find();]
[> db.donationsdb.find();]
[> db.donationsdb.insert({ "paymenttype": "PayPal", "amount": 1000, "upvotes": 0 })
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.insert({ "paymenttype": "Direct", "amount": 1500, "upvotes": 0 })
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.find({ "_id" : "566593af87282d5d60eedc21" })
[> db.donationsdb.find({ "_id" : ObjectId("566593af87282d5d60eedc21") })
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
[> db.donationsdb.insert({ "paymenttype": "PayPal", "amount": 1500, "upvotes": 0 })
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.find({ "amount": 1500 })
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
>

```



# Delete / Remove a Record

```

ddrohan — mongo — 121×24
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.insert({"paymenttype":"Direct", "amount":1500, "upvotes":0})
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.find({_id : "566593af87282d5d60eedc21"})
[> db.donationsdb.find({_id : ObjectId("566593af87282d5d60eedc21")})
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
[> db.donationsdb.insert({"paymenttype":"PayPal", "amount":1500, "upvotes":0})
WriteResult({ "nInserted" : 1 })
[> db.donationsdb.find({"amount":1500})
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.remove({_id : ObjectId("566593af87282d5d60eedc21")})
WriteResult({ "nRemoved" : 1 })
[> db.donationsdb.find()
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
>

```

# Update a Record

```
[> db.donationsdb.find({"amount":1500})
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.find()
{ "_id" : ObjectId("566593af87282d5d60eedc21"), "paymenttype" : "PayPal", "amount" : 1000, "upvotes" : 0 }
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.remove({"_id" : ObjectId("566593af87282d5d60eedc21")})
WriteResult({ "nRemoved" : 1 })
[> db.donationsdb.find()
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 0 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
[> db.donationsdb.update({ "_id" : ObjectId("566593d087282d5d60eedc22"), { $set: "upvotes" : 1 }})
2015-12-07T15:02:54.316+0000 E QUERY    SyntaxError: Unexpected token {
[> db.donationsdb.update({ "_id" : ObjectId("566593d087282d5d60eedc22") }, { $set: "upvotes" : 1 })
2015-12-07T15:02:54.893+0000 E QUERY    SyntaxError: Unexpected token :
[> db.donationsdb.update({ "_id" : ObjectId("566593d087282d5d60eedc22") }, { $set: { "upvotes" : 1 }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
[>
[> db.donationsdb.find()
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 1 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
>
```

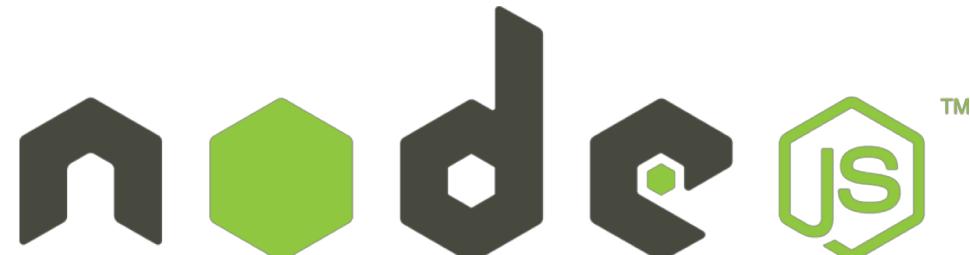
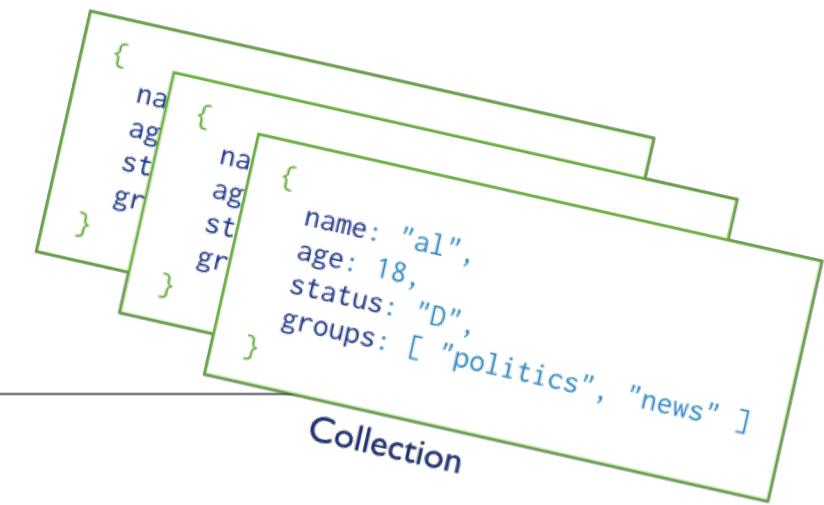
# MongoDB & Node.js

---

OBJECT-DOCUMENT MODEL



{ name: mongo, type: DB }



# MongoDB Queries

---

The MongoDB module supports all kinds of queries over the data

- Creating new documents
  - And adding records
- Editing existing documents
  - And their records
- Removing documents and records
- Querying whole documents or parts of them

# Using MongoDB with Node.js

---

Once installed, MongoDB must be started before it can be used

- Go to installation folder and run `mongod`

```
$ cd path/to/mongodb/installation/folder  
$ mongod
```

- Or add `mongod.exe` to the PATH

When run, MongoDB can be used from Node.js

# Using MongoDB with Node.js

---

Download MongoDB from the official web site:

- <https://www.mongodb.org/downloads>
- Installers for all major platforms

When installed, MongoDB needs a driver to be usable with a specific platform

- One to use with Node.js, another to use with .NET, etc...

Installing MongoDB dependency module for Node.js:

```
$ npm install mongodb -g
```

# MongoDB Module & Node.js

---

- The 'mongodb' module is required

```
var mongodb = require('mongodb');
```

- Create a server to host the database

```
var server = new mongodb.Server('localhost', 27017);
```

- Create mongodb client that connects to the server

```
var mongoClient = new mongodb.MongoClient(server);
```

- Open connection to the mongodb server

```
mongoClient.open(function(err, client){  
    var db = client.db('DATABASE_NAME');  
    //queries over the db  
});
```

# mongoose

elegant `mongodb` object modeling for `node.js`

# Mongoose Overview

---

OBJECT-DOCUMENT MODEL MODULE FOR NODE.JS

# Mongoose Overview

---

Mongoose is a object-document model module in Node.js for MongoDB

- Wraps the functionality of the native MongoDB driver
- Exposes models to control the records in a doc
- Supports validation on save
- Extends the native queries

# Installing Mongoose

---

Run the following from the CMD/Terminal

```
$ npm install mongoose
```

Load the Module

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

Connect to the Database

```
mongoose.connect(mongoDbPath);
```

Create Models and persist data

```
var Unit = mongoose.model('Unit', { type: String } );
new Unit({type: 'warrior'}).save(callback); //create
Unit.find({type: 'warrior'}).exec(callback); //fetch
```

# Mongoose Models

---

OBJECT-DOCUMENT MODEL SCHEMA

# mongoose

elegant `mongodb` object modeling for `node.js`

```
let mongoose = require('mongoose');
let mongoose = require('mongoose');
let mongoose = require('mongoose');
let mongoose = require('mongoose');
let DonationSchema = new mongoose.Schema({
  paymenttype: String,
  amount: Number,
  upvotes: {type: Number, default: 0},
  collection: 'donationsdb' });
module.exports = mongoose.model('Donation', DonationSchema);
```

# Mongoose Models

---

- ❑ Mongoose supports Models
  - i.e. fixed types of documents
- ❑ Used like object Constructors
  - Needs a ‘mongoose.Schema’

```
let mongoose = require('mongoose');

let DonationSchema = new mongoose.Schema({
  paymenttype: String,
  amount: Number,
  upvotes: {type: Number, default: 0}
},
{ collection: 'donationsdb' });

module.exports = mongoose.model('Donation', DonationSchema);
```

# Mongoose Models

---

- ❑ Each of the properties must have a type
  - ❑ Types can be Number, String, Boolean, array, object
  - ❑ Even nested objects
  - ❑ Good idea to explicitly name your collection

```
let mongoose = require('mongoose');

let DonationSchema = new mongoose.Schema({
  paymenttype: String,
  amount: Number,
  upvotes: {type: Number, default: 0}
},
{ collection: 'donationsdb' });

module.exports = mongoose.model('Donation', DonationSchema);
```

# mongoose

elegant `mongodb` object modeling for `node.js`

# CRUD with Mongoose

---

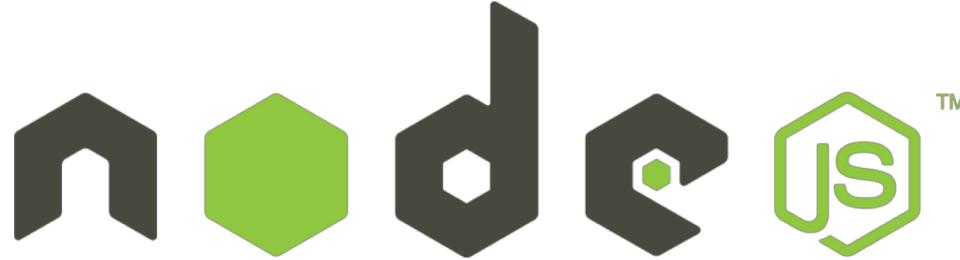
OBJECT-DOCUMENT MODEL SCHEMA

```
let mongoose = require('mongoose');
let mongoose = require('mongoose');
let mongoose = require('mongoose');
let mongoose = require('mongoose');
let DonationSchema = new mongoose.Schema({
  paymenttype: String,
  amount: Number,
  upvotes: {type: Number, default: 0}
}, {
  collection: 'donationsdb'
});
module.exports = mongoose.model('Donation', DonationSchema);
```

# CRUD with Mongoose

---

- ❑ Mongoose supports all the CRUD operations:
  - ❑ Create -> `modelObj.save(callback)`
  - ❑ Read -> `Model.find().exec(callback)`
  - ❑ Update -> `modelObj.update(props, callback)`  
                 -> `Model.update(condition, props, callback)`
  - ❑ Remove -> `modelObj.remove(callback)`  
                 -> `Model.remove(condition, props, callback)`

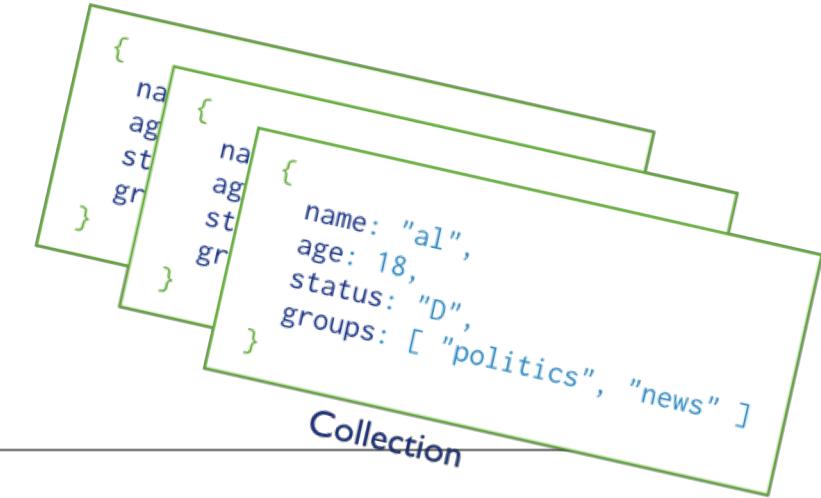


# donationweb

CASE STUDY EXAMPLE



{ name: mongo, type: DB }



# mongoose

elegant [mongodb](#) object modeling for [node.js](#)

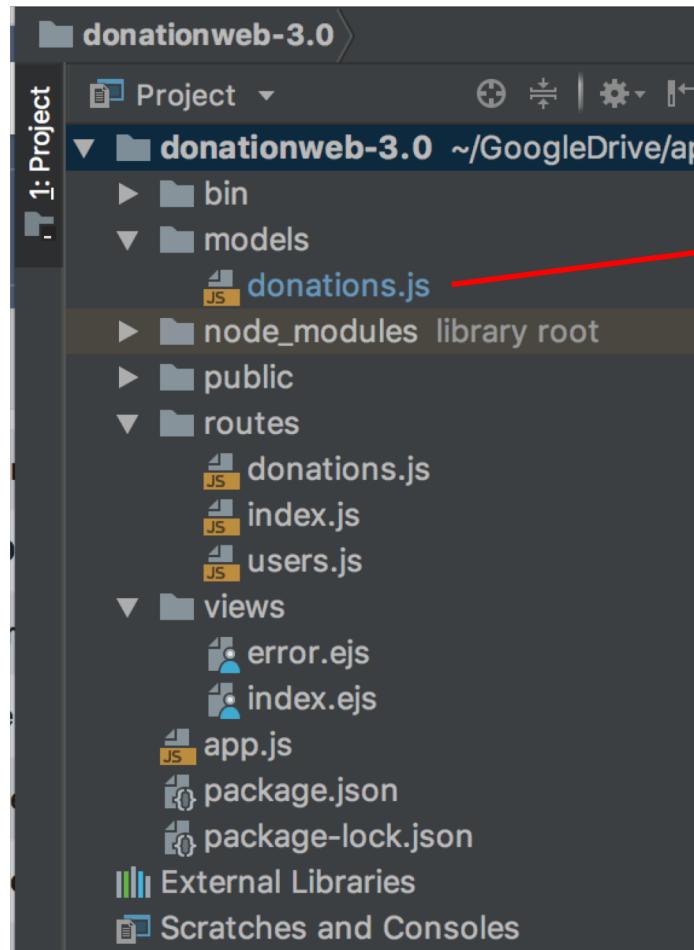
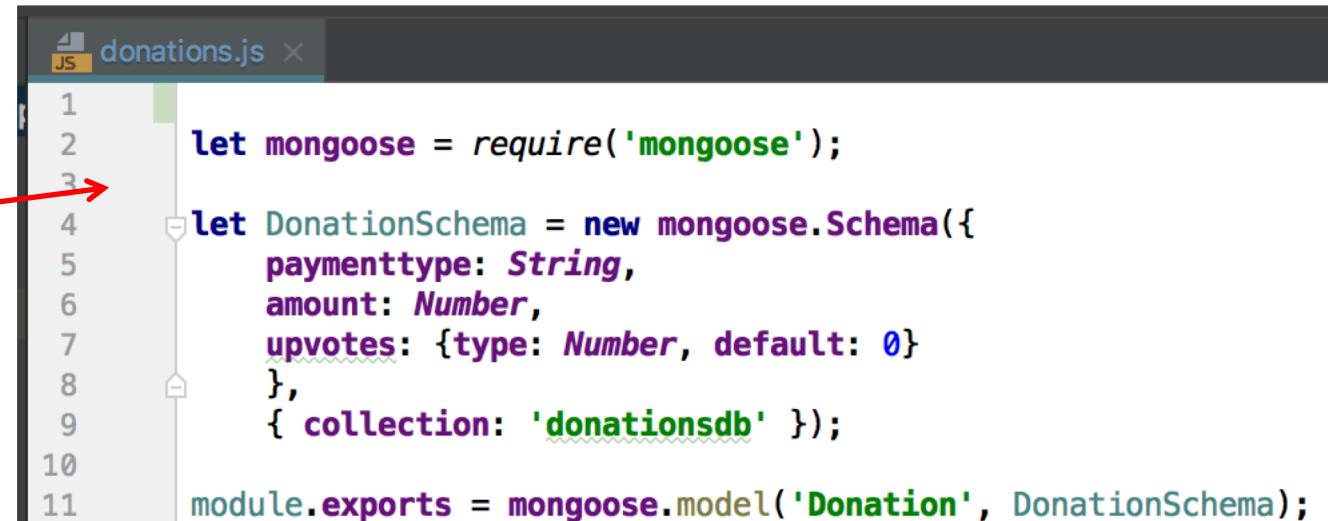
# Donation: Resource, URIs & Methods

Resource	URI (structure)	HTTP Request
List of Donations	/donations	GET
Get a Single Donation	/donations/{id}	GET
Upvote a Donation	/donations/{id}/vote	PUT
Delete a Donation	/donations/{id}	DELETE
Update a Donation	/donations/{id}	PUT
Add a Donation	/donations/{id}	POST
Total of Donation Votes	/donations/votes	GET

{...} = variable value; changeable by user/application to refer to specific resource

We'll look at this Use Case as an example...

# Creating the Model – *Server Side*

```

1  let mongoose = require('mongoose');
2
3  let DonationSchema = new mongoose.Schema({
4      paymenttype: String,
5      amount: Number,
6      upvotes: {type: Number, default: 0}
7  },
8  { collection: 'donationsdb' });
9
10
11 module.exports = mongoose.model('Donation', DonationSchema);

```

# Creating the Routes (1) – Server Side

donationweb-3.0

Project ~/GoogleDrive/ap

- donationweb-3.0
  - bin
  - models
    - donations.js
  - node\_modules library root
  - public
  - routes
    - donations.js
    - index.js
    - users.js
  - views
    - error.ejs
    - index.ejs
  - app.js
  - package.json
  - package-lock.json
- External Libraries
- Scratches and Consoles

donations.js

```

1 let Donation = require('../models/donations');
2 let express = require('express');
3 let router = express.Router();
4 let mongoose = require('mongoose');
5 let uriUtil = require('mongodb-uri');
6
7 var mongoDBUri = 'mongodb://[REDACTED]:[REDACTED]@ds255260.mlab.com:55260/donationsdb';
8
9 mongoose.connect(mongoDBUri);
10
11 //mongoose.connect('mongodb://localhost:27017/donationsdb');
12
13 let db = mongoose.connection;
14
15 db.on('error', function (err) {
16   console.log('Unable to Connect to [ ' + db.name + ' ]', err);
17 });
18
19 db.once('open', function () {
20   console.log('Successfully Connected to [ ' + db.name + ' ] on mlab.com');
21 });
  
```

N.B. on 'imports'

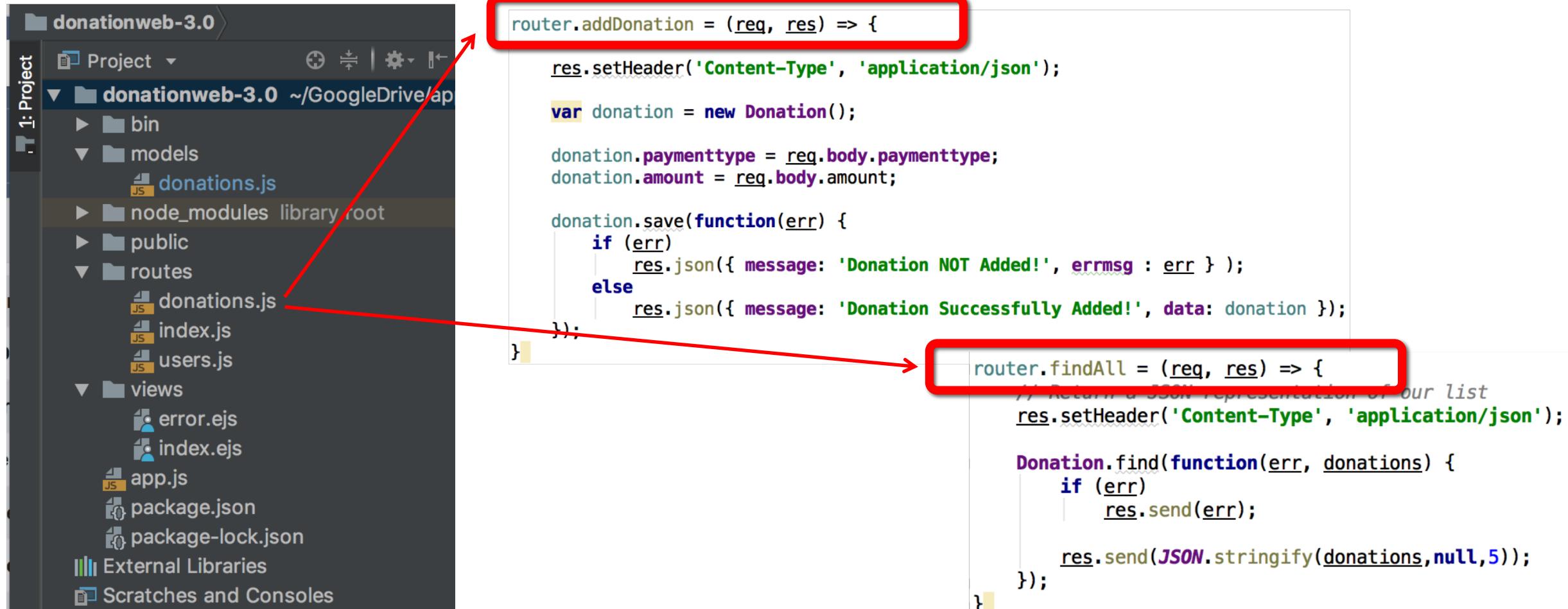
# Creating the Routes (2) – Server Side

donationweb-3.0

```

1: Project
  └── donationweb-3.0 ~/GoogleDrive/ap
    ├── bin
    ├── models
    │   └── donations.js
    ├── node_modules library root
    ├── public
    └── routes
      ├── donations.js
      ├── index.js
      └── users.js
  └── views
    ├── error.ejs
    ├── index.ejs
    ├── app.js
    ├── package.json
    └── package-lock.json
  └── External Libraries
  └── Scratches and Consoles

```



```

router.addDonation = (req, res) => {
  res.setHeader('Content-Type', 'application/json');

  var donation = new Donation();

  donation.paymenttype = req.body.paymenttype;
  donation.amount = req.body.amount;

  donation.save(function(err) {
    if (err)
      res.json({ message: 'Donation NOT Added!', errmsg : err });
    else
      res.json({ message: 'Donation Successfully Added!', data: donation });
  });
};

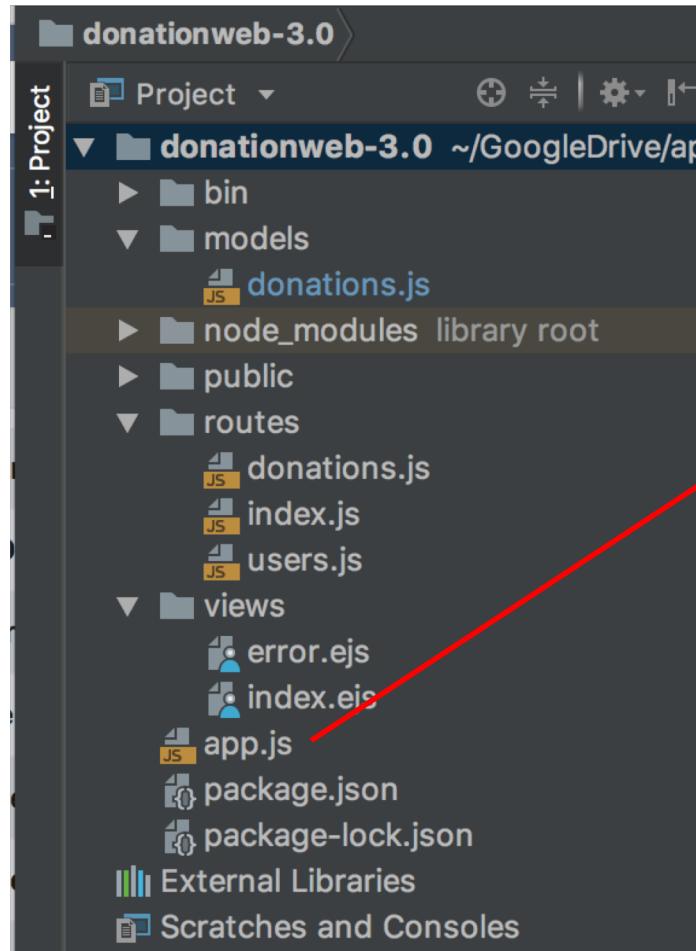
router.findAll = (req, res) => {
  // Return a JSON representation of our list
  res.setHeader('Content-Type', 'application/json');

  Donation.find(function(err, donations) {
    if (err)
      res.send(err);

    res.send(JSON.stringify(donations,null,5));
  });
};

```

# Creating the Routes (3) – Server Side



```
// Our Custom Donation Web App Routes
app.get('/donations', donations.findAll);
app.get('/donations/votes', donations.findTotalVotes);
app.get('/donations/:id', donations.findOne);

app.post('/donations', donations.addDonation);

app.put('/donations/:id/vote', donations.incrementUpvotes);

app.delete('/donations/:id', donations.deleteDonation);
```

# Great Resources

---

Official Tutorial – <https://nodejs.org/documentation/tutorials/>

Official API – <https://nodejs.org/api/>

Developer Guide – <https://nodejs.org/documentation>

Video Tutorials – <http://nodetuts.com>

Video Introduction – <https://www.youtube.com/watch?v=FqMlyTH9wSg>

YouTube Channel – [https://www.youtube.com/channel/UCvhlsEIBIfWSn\\_Fod8FuuGg](https://www.youtube.com/channel/UCvhlsEIBIfWSn_Fod8FuuGg)

Articles, explanations, tutorials – <https://nodejs.org/community/>

# Questions?