

# Dynamic Web Development

---

Produced  
by

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

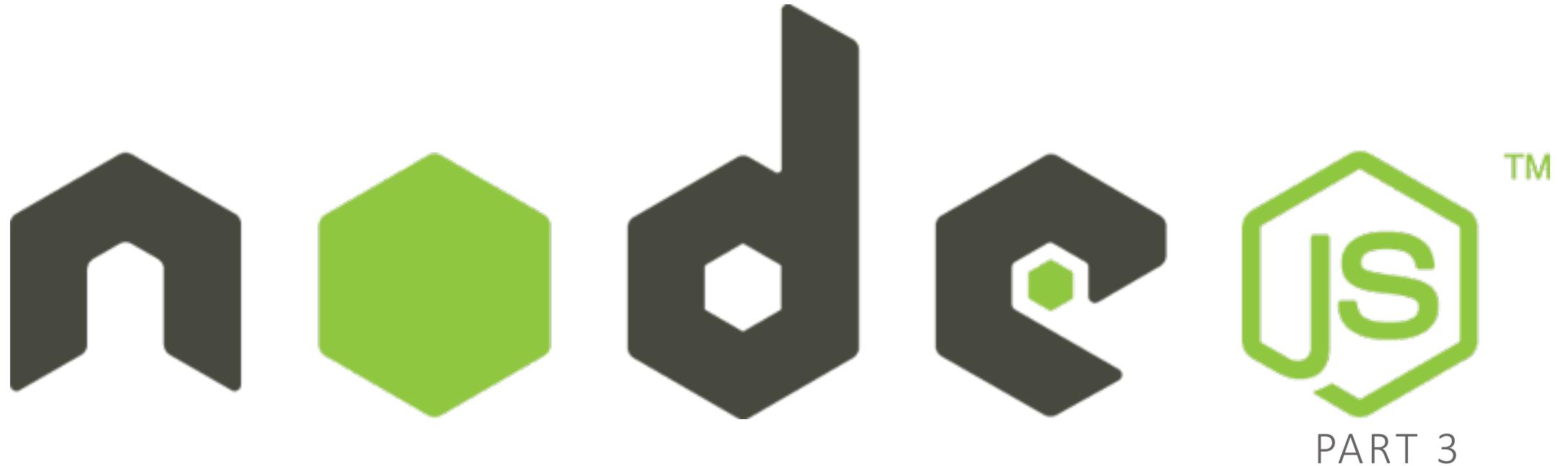
Department of Computing & Mathematics  
Waterford Institute of Technology

<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



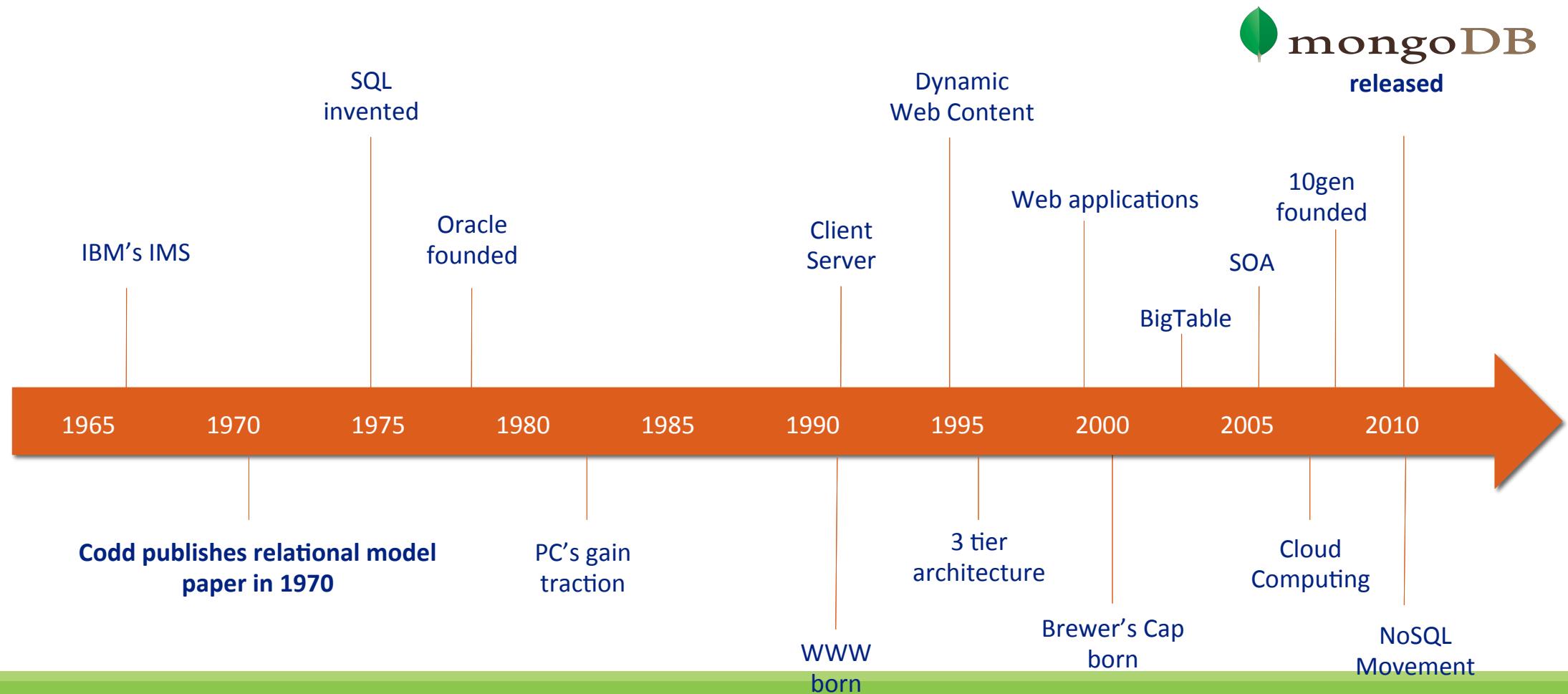
# Databases

---

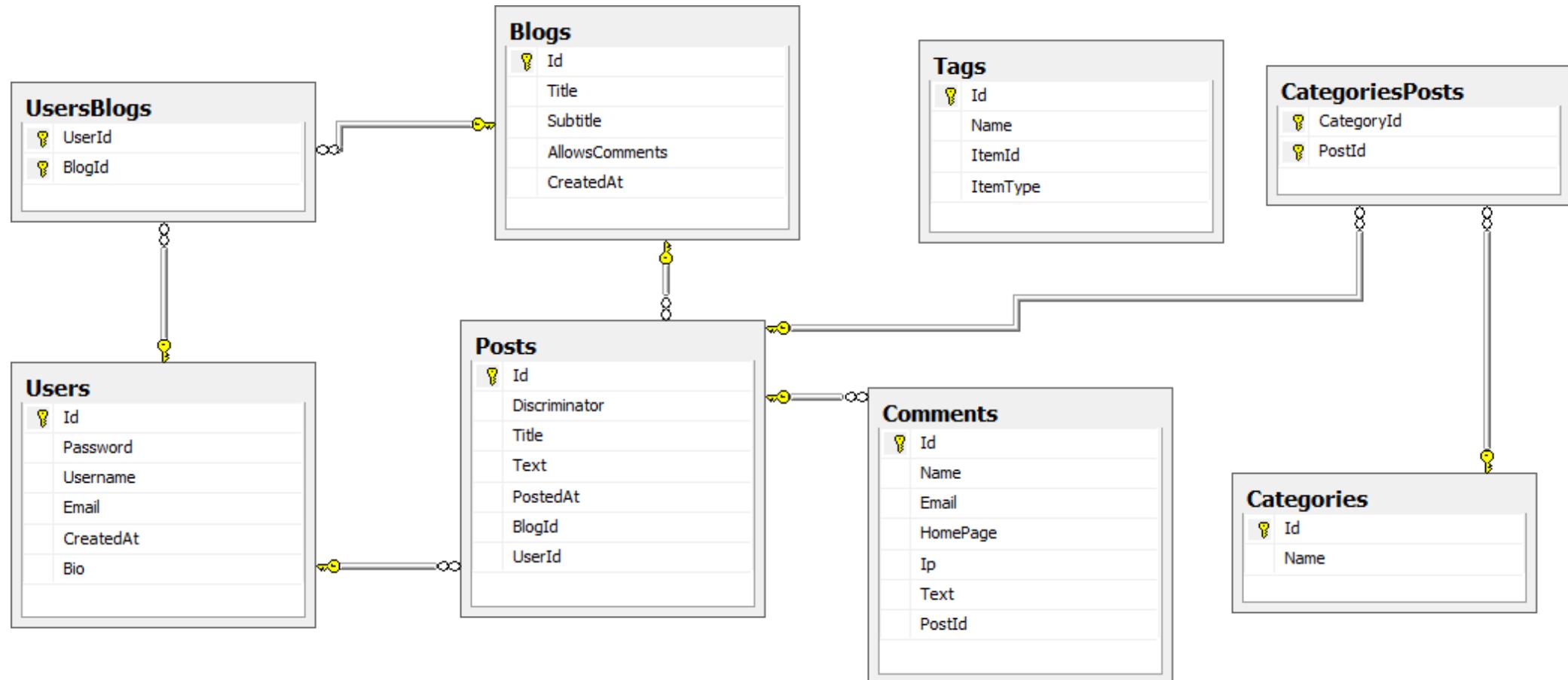
HISTORY & BACKGROUND



# Dawn of Databases to Present



# 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

---

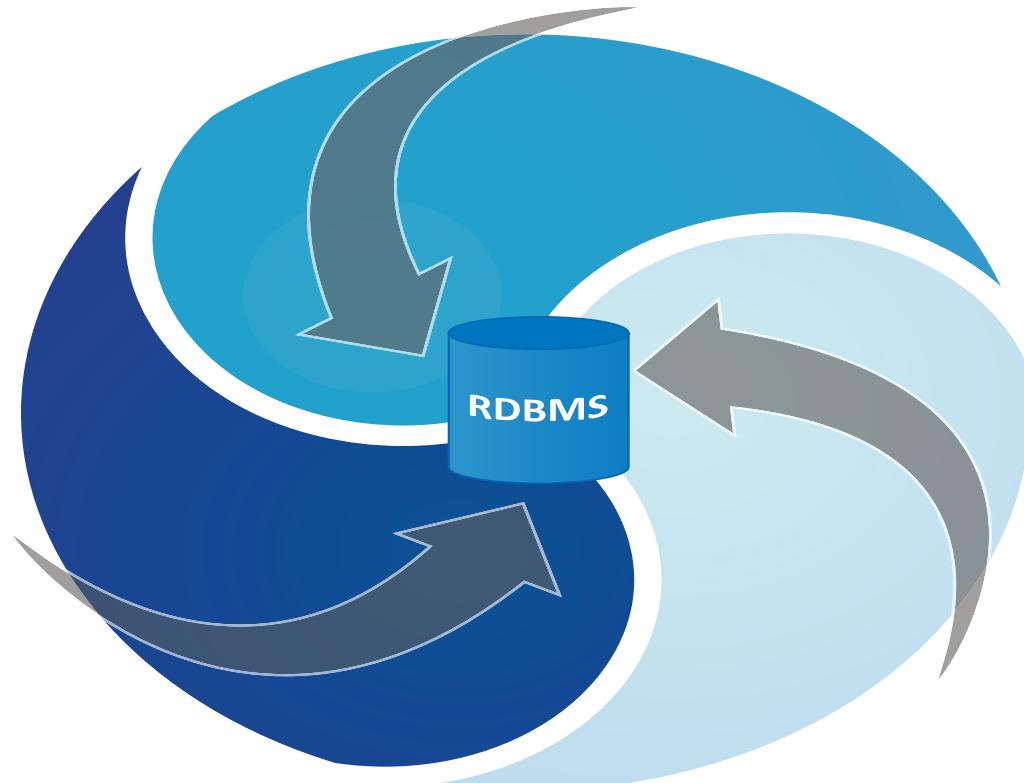
Gartner uses the 3Vs to define

- 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 (Dec 2015)

289 systems in ranking, December 2015

Rank			DBMS	Database Model	Score		
Dec 2015	Nov 2015	Dec 2014			Dec 2015	Nov 2015	Dec 2014
1.	1.	1.	Oracle	Relational DBMS	1497.55	+16.61	+37.76
2.	2.	2.	MySQL	Relational DBMS	1298.54	+11.70	+29.96
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1123.16	+0.83	-76.89
4.	4.	↑ 5.	MongoDB +	Document store	301.39	-3.22	+54.87
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	280.09	-5.60	+26.09
6.	6.	6.	DB2	Relational DBMS	196.13	-6.40	-14.13
7.	7.	7.	Microsoft Access	Relational DBMS	140.21	-0.75	+0.31
8.	8.	↑ 9.	Cassandra +	Wide column store	130.84	-2.08	+36.78
9.	9.	↓ 8.	SQLite	Relational DBMS	100.85	-2.60	+6.15
10.	10.	10.	Redis +	Key-value store	100.54	-1.87	+12.66

# Overview: MongoDB Key Features

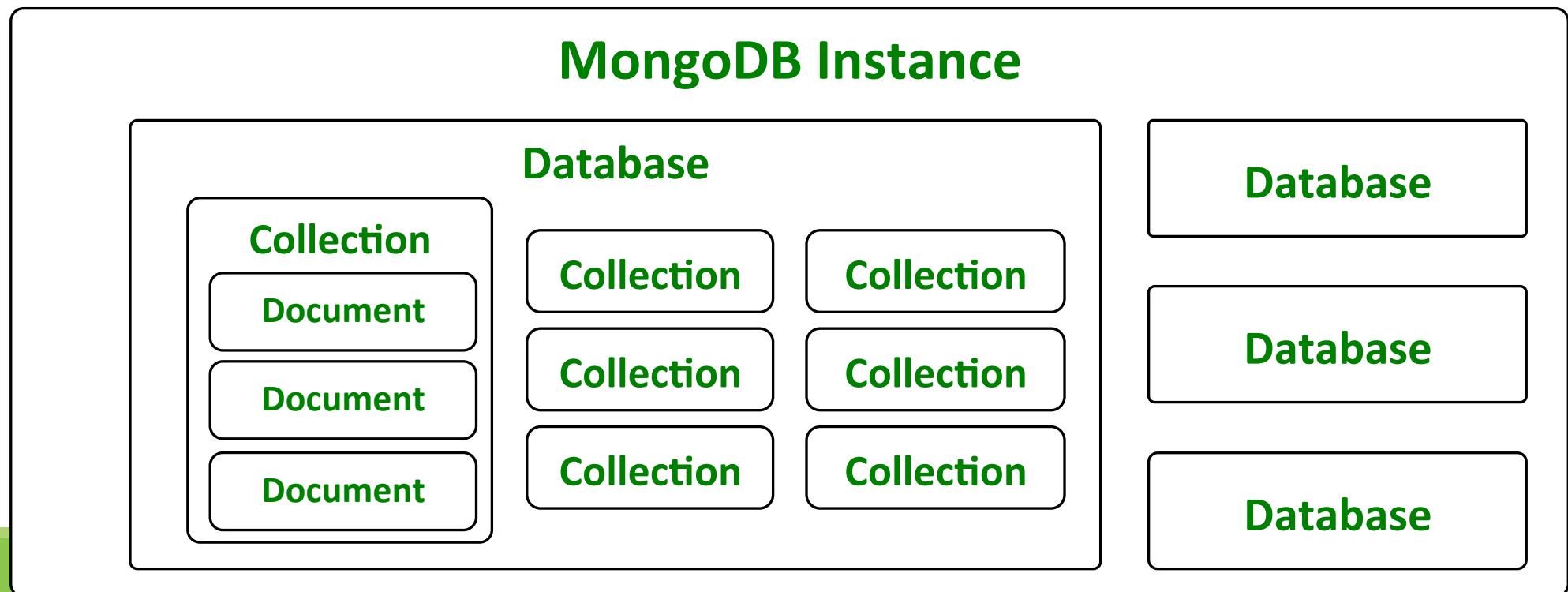
---

- 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 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

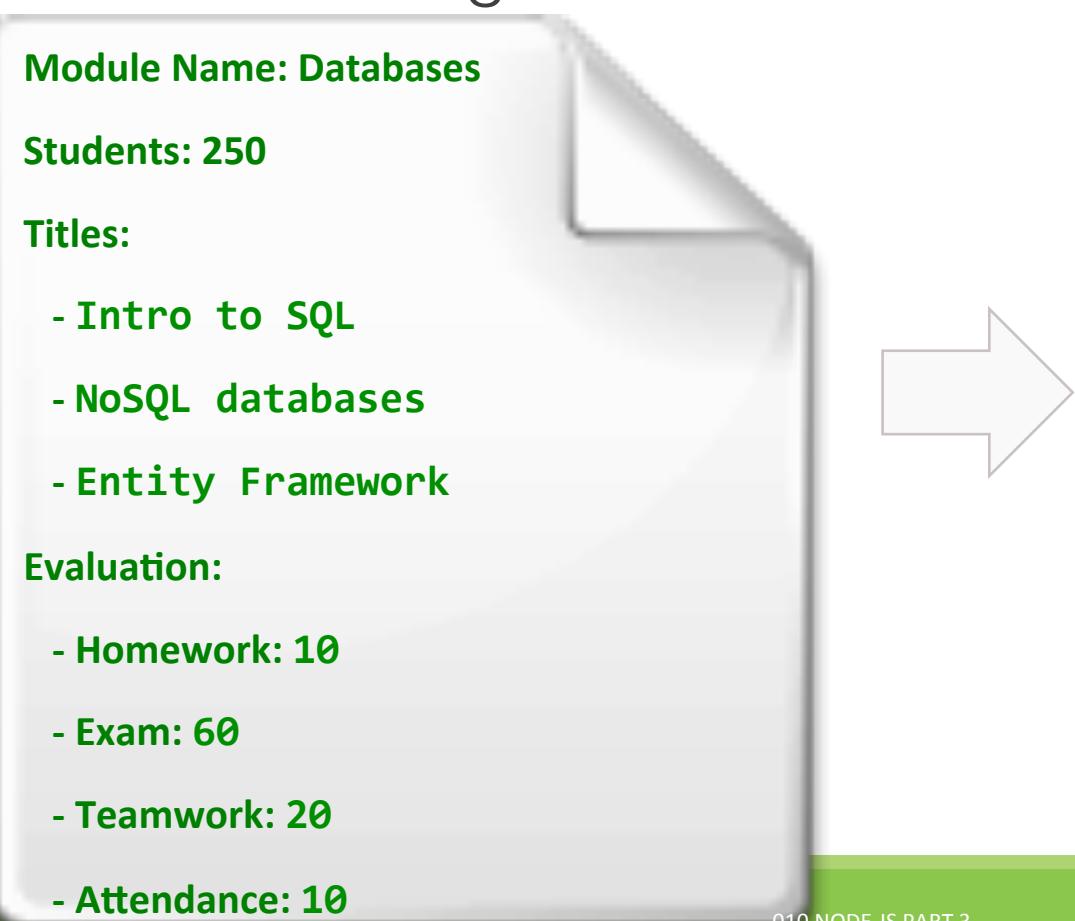
**Students:** 250

**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": "daveyD",  
  "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

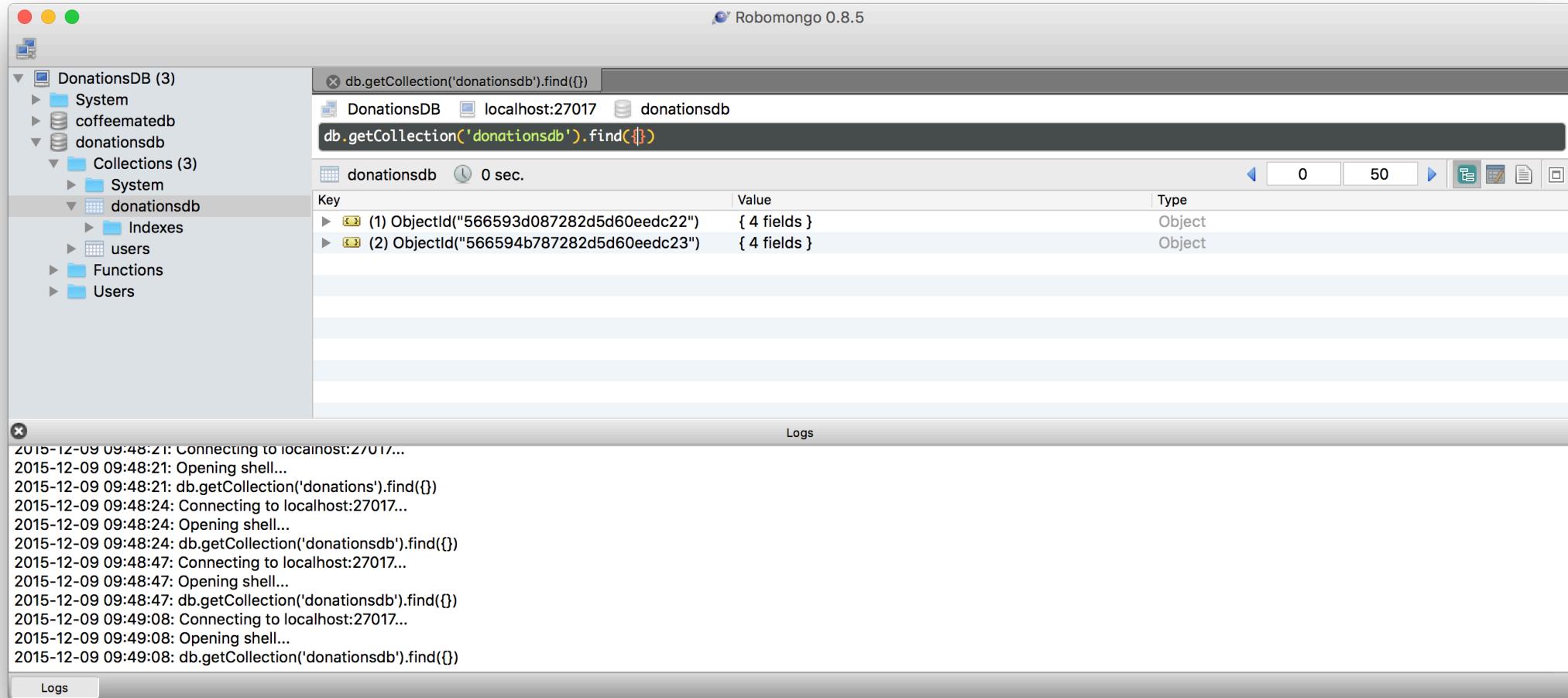
- So there are many available viewers

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 interface. On the left is a tree view of databases and collections:

- DonationsDB (3)
  - System
  - coffeematedb
  - donationsdb
    - Collections (3)
      - System
      - donationsdb
        - Indexes
        - users
      - Functions
      - Users

In the main pane, a query builder window is open with the command `db.getCollection('donationsdb').find({})`. The results show two documents in the `donationsdb` collection:

Key	Value	Type
<code>(1) ObjectId("566593d087282d5d60eedc22")</code>	{ 4 fields }	Object
<code>(2) ObjectId("566594b787282d5d60eedc23")</code>	{ 4 fields }	Object

At the bottom, a logs pane displays the following entries:

```

2015-12-09 09:48:21: Connecting to localhost:27017...
2015-12-09 09:48:21: Opening shell...
2015-12-09 09:48:21: db.getCollection('donations').find({})
2015-12-09 09:48:24: Connecting to localhost:27017...
2015-12-09 09:48:24: Opening shell...
2015-12-09 09:48:24: db.getCollection('donationsdb').find({})
2015-12-09 09:48:47: Connecting to localhost:27017...
2015-12-09 09:48:47: Opening shell...
2015-12-09 09:48:47: db.getCollection('donationsdb').find({})
2015-12-09 09:49:08: Connecting to localhost:27017...
2015-12-09 09:49:08: Opening shell...
2015-12-09 09:49:08: db.getCollection('donationsdb').find({})

```

# MongoDB - Robomongo

Robomongo 0.8.5

DonationsDB (3)

- System
- coffeematedb
- ▼ donationsdb
  - Collections (3)
  - System
  - donationsdb
    - Indexes
    - users
  - Functions
  - Users

db.getCollection('donationsdb').find({})

DonationsDB
localhost:27017
donationsdb

```
db.getCollection('donationsdb').find({})
```

donationsdb 0 sec.

Key	Value	Type
► (1) ObjectId("566593d087282d5d60eedc22")	{ 4 fields }	Object
► _id	ObjectId("566593d087282d5d60eedc22")	Objectid
► paymenttype	Direct	String
► amount	1500.000000	Double
► upvotes	1.000000	Double
► (2) ObjectId("566594b787282d5d60eedc23")	{ 4 fields }	Object

Logs

```
2015-12-09 09:48:21: Connecting to localhost:27017...
2015-12-09 09:48:21: Opening shell...
2015-12-09 09:48:21: db.getCollection('donations').find({})
2015-12-09 09:48:24: Connecting to localhost:27017...
2015-12-09 09:48:24: Opening shell...
2015-12-09 09:48:24: db.getCollection('donationsdb').find({})
2015-12-09 09:48:47: Connecting to localhost:27017...
2015-12-09 09:48:47: Opening shell...
2015-12-09 09:48:47: db.getCollection('donationsdb').find({})
2015-12-09 09:49:08: Connecting to localhost:27017...
2015-12-09 09:49:08: Opening shell...
2015-12-09 09:49:08: db.getCollection('donationsdb').find({})
```

# Donationweb

---

CASE STUDY EXAMPLE



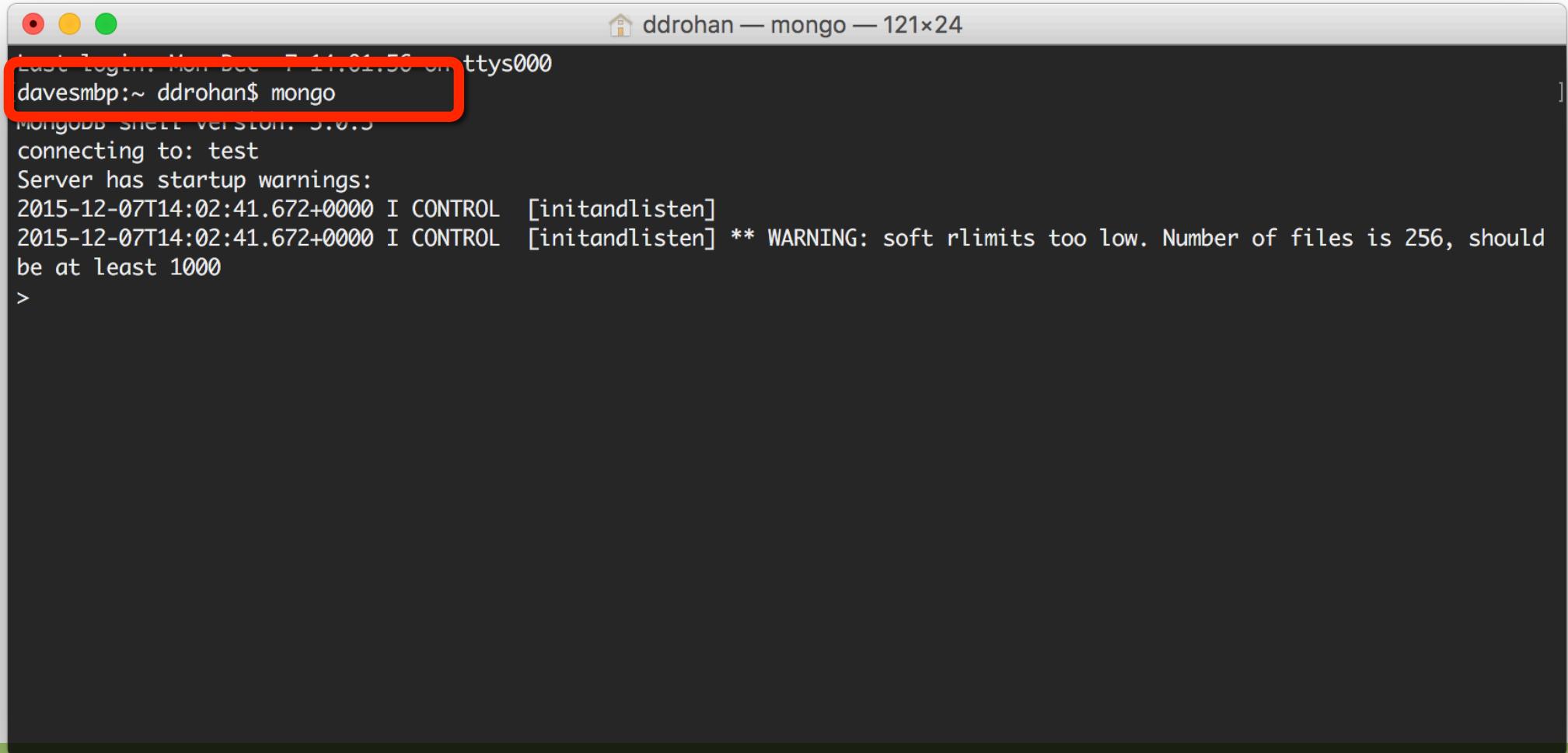
{ name: mongo, type: DB }



# Launch the MongoDB Server

```
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



```
ddrohan@davesmbp:~$ 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
coffeematedb 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
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("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})
```

The last command, `db.donationsdb.insert({ "paymenttype": "PayPal", "amount": 1500, "upvotes": 0 })`, is highlighted with a red box.

```

[> 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 — 121x24
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 })
2015-12-07T15:06:54.893+0000 E QUERY    SyntaxError: Unexpected token :
[> db.donationsdb.update({ "_id" : ObjectId("566593d087282d5d60eedc22") }, { $set: { "upvotes" : 1 }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

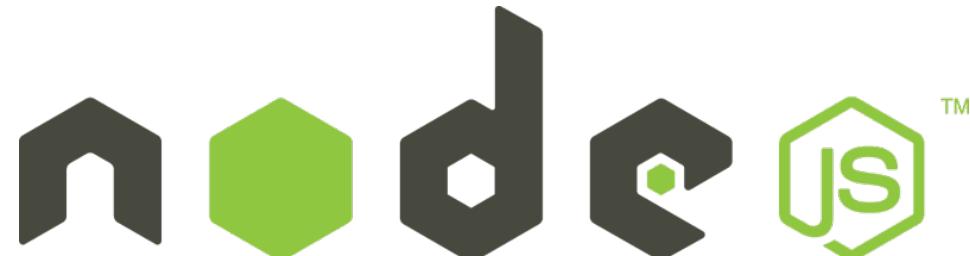
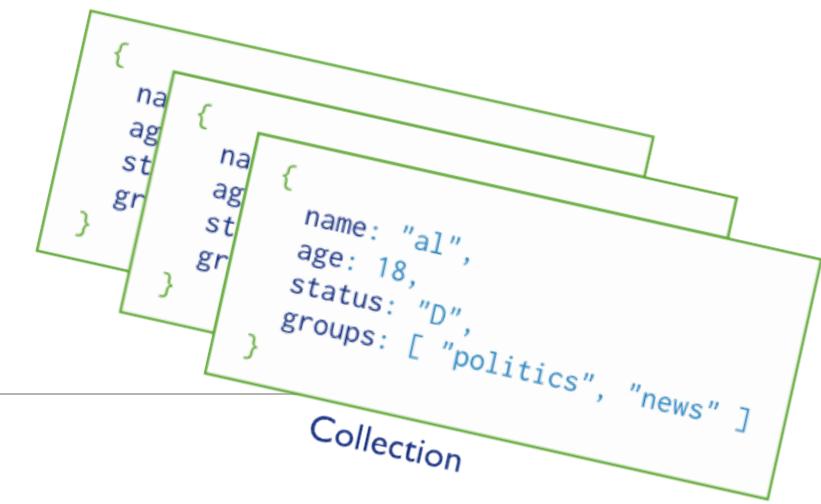
```
[> db.donationsdb.find()
{ "_id" : ObjectId("566593d087282d5d60eedc22"), "paymenttype" : "Direct", "amount" : 1500, "upvotes" : 1 }
{ "_id" : ObjectId("566594b787282d5d60eedc23"), "paymenttype" : "PayPal", "amount" : 1500, "upvotes" : 0 }
>
```

# 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
```

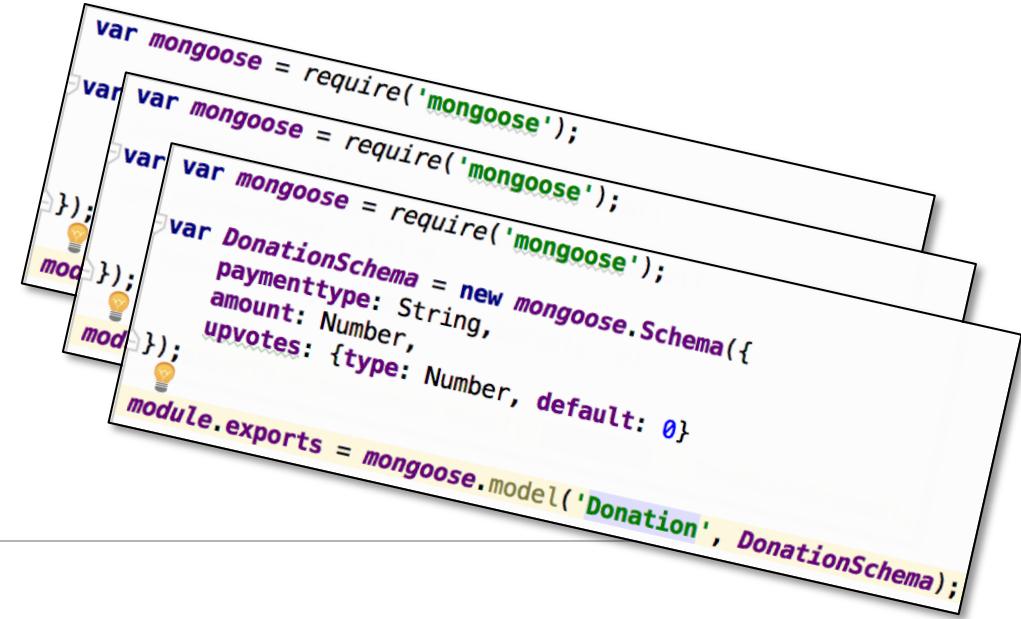
# Mongoose Models

---

OBJECT-DOCUMENT MODEL SCHEMA

# mongoose

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



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

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

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’

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

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

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

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

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

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



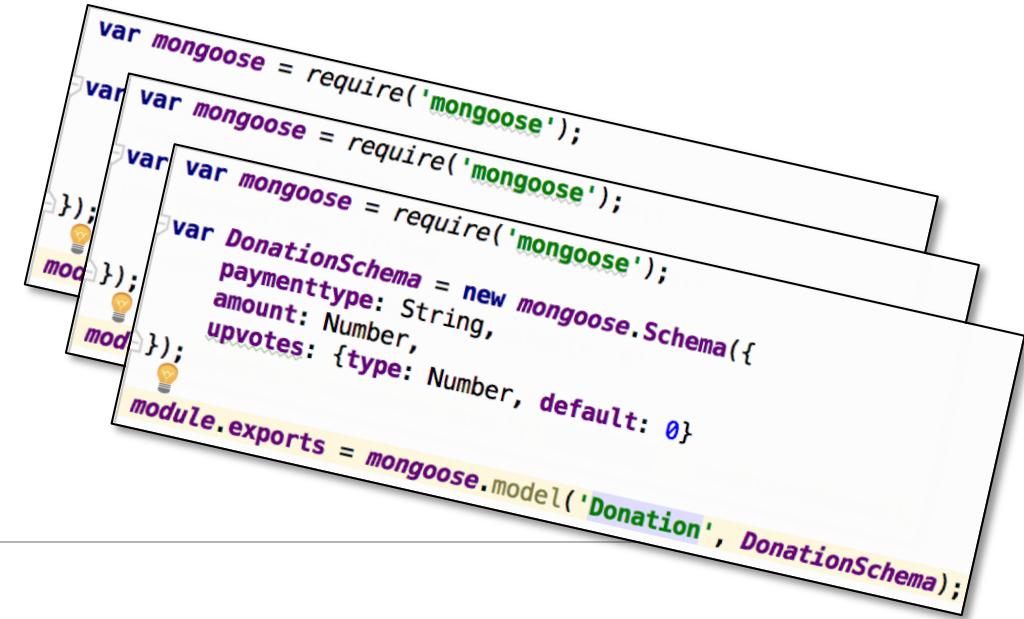
# mongoose

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

# CRUD with Mongoose

---

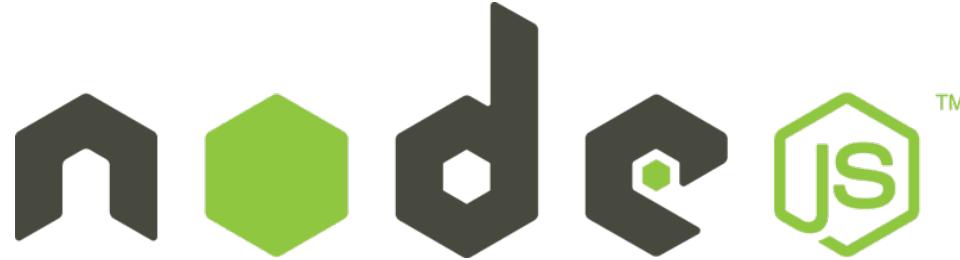
OBJECT-DOCUMENT MODEL SCHEMA



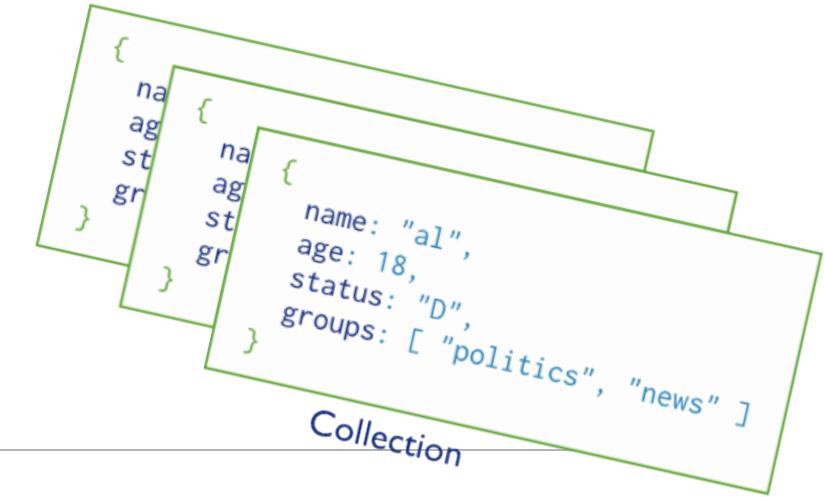
# CRUD with Mongoose

---

- ◆ 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)`



{ name: mongo, type: DB }



# Donationweb

---

CASE STUDY EXAMPLE

# mongoose

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

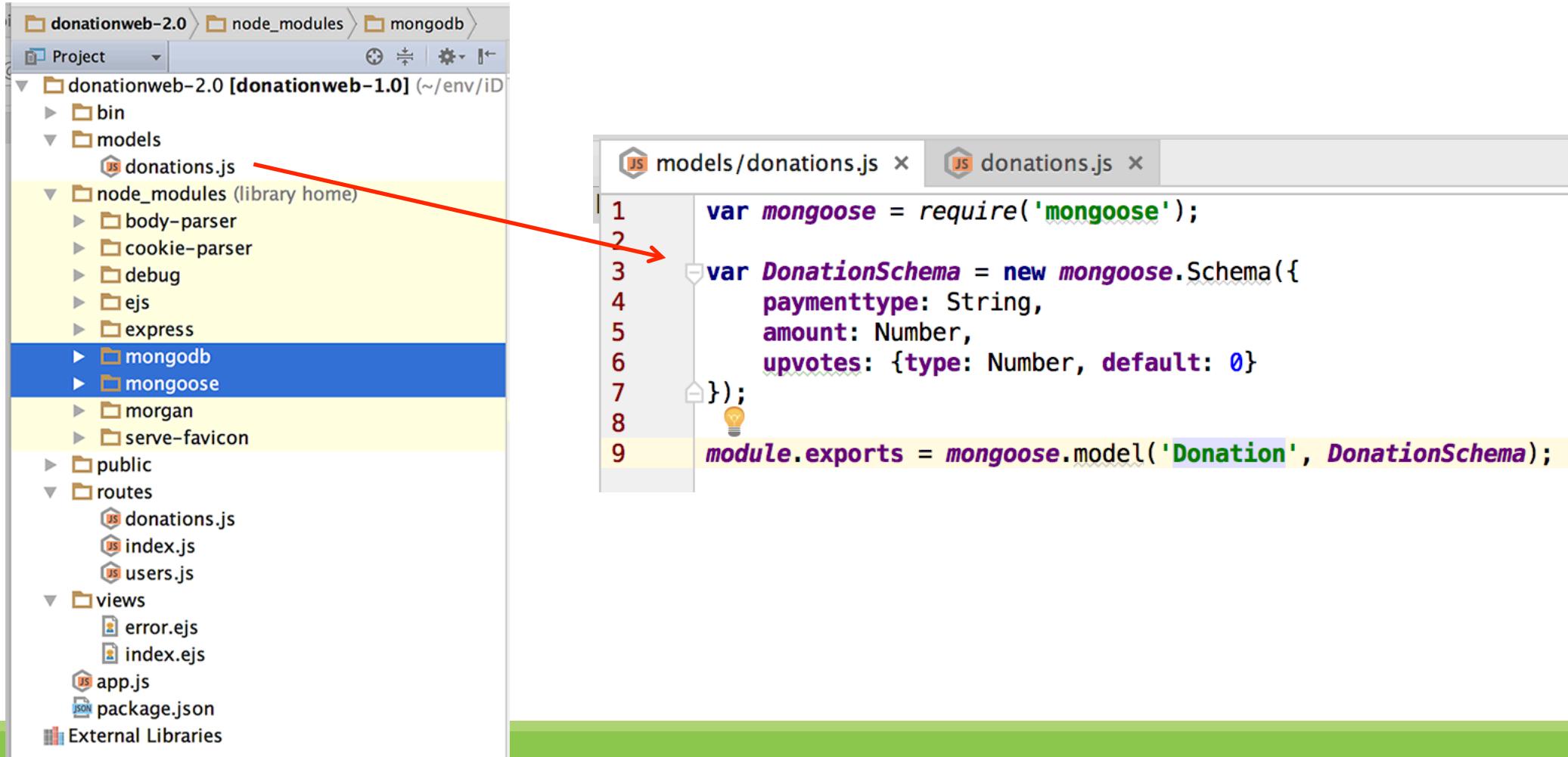
# Donation: Resource, URIs & Methods

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

{...} = 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*



The screenshot shows a file structure on the left and a code editor on the right.

**File Structure:**

- donationweb-2.0 > node\_modules > mongodb
- donationweb-2.0 [donationweb-1.0] (~env/iD)
- models
  - donations.js
- node\_modules (library home)
  - body-parser
  - cookie-parser
  - debug
  - ejs
  - express
  - mongodb
  - mongoose
  - morgan
  - serve-favicon
- public
- routes
  - donations.js
  - index.js
  - users.js
- views
  - error.ejs
  - index.ejs
- app.js
- package.json
- External Libraries

A red arrow points from the 'models' folder in the file structure to the 'donations.js' file in the code editor.

**Code Editor:**

```

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

The code defines a Mongoose schema for a 'Donation' model. It includes fields for 'paymenttype' (String), 'amount' (Number), and 'upvotes' (Number, defaulting to 0). The schema is then exported as 'module.exports'.

# Creating the Routes (1) – Server Side

The image shows a code editor and a file explorer side-by-side.

**File Explorer:**

- Project: donationweb-2.0 [donationweb-1.0] (~/env/iD)
- models/donations.js
- node\_modules (library home):
  - body-parser
  - cookie-parser
  - debug
  - ejs
  - express
  - mongodb
  - mongoose
  - morgan
  - serve-favicon
- public
- routes/donations.js
- views/error.ejs
- views/index.ejs
- app.js
- package.json
- External Libraries

A red arrow points from the 'routes' folder in the file explorer to the 'routes/donations.js' file in the code editor.

**Code Editor:**

```

1 var Donation = require('../models/donations');
2 var express = require('express');
3 var router = express.Router();
4 var mongoose = require('mongoose');

5
6
7 mongoose.connect('mongodb://localhost:27017/donationsdb');

8 var db = mongoose.connection;
9
10 db.on('error', function (err) {
11   console.log('connection error', err);
12 });
13 db.once('open', function () {
14   console.log('connected to database');
15 });
16
17
18 router.findAll = function(req, res) {...}
19
20 router.findOne = function(req, res) {...}
21
22 router.addDonation = function(req, res) {...}
23
24 router.deleteDonation = function(req, res) {...}
25
26 router.incrementUpvotes = function(req, res) {...}

27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84 module.exports = router;
  
```

A red arrow points from the 'mongoose' import statement in the code to the 'mongoose' entry in the file explorer's node\_modules list. Another red arrow points from the 'mongoose' import statement to the text 'N.B. on "imports"' located to its right.

**N.B. on 'imports'**

# Creating the Routes (2) – Server Side

File structure:

```

donationweb-2.0 > node_modules > mongodb >
  Project
  donationweb-2.0 [donationweb-1.0] (~env/iD)
    bin
    models
      donations.js
    node_modules (library home)
      body-parser
      cookie-parser
      debug
      ejs
      express
      mongodb
      mongoose
      morgan
      serve-favicon
    public
    routes
      donations.js
      index.js
      users.js
    views
      error.ejs
      index.ejs
    app.js
    package.json
  External Libraries

```

Code snippets:

```

// In routes/donations.js
router.addDonation = function(req, res) {
  var donation = new Donation();
  donation.paymenttype = req.body.paymenttype;
  donation.amount = req.body.amount;
  //console.log('Adding donation: ' + JSON.stringify(donation));
  // Save the donation and check for errors
  donation.save(function(err) {
    if (err)
      res.send(err);
    res.json({ message: 'Donation Added!', data: donation });
  });
};

// In routes/index.js
router.findAll = function(req, res) {
  // Use the Donation model to find all donations
  Donation.find(function(err, donations) {
    if (err)
      res.send(err);
    else
      res.json(donations);
  });
};

```

# Creating the Routes (3) – *Server Side*

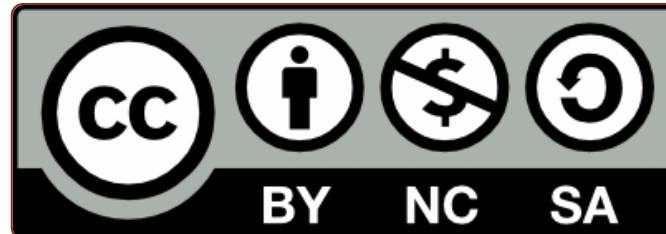
The screenshot shows a code editor with the file `app.js` open. A red arrow points from the `mongodb` folder in the project tree on the left to the `require('mongodb')` line in the code editor. The project tree also includes `node_modules`, `models` (with `donations.js`), `routes` (with `donations.js`, `index.js`, `users.js`), and `views` (with `error.ejs`, `index.ejs`, `app.js`, and `package.json`). The `External Libraries` section is also visible.

```
1 var express = require('express');
2 var path = require('path');
3 var favicon = require('serve-favicon');
4 var logger = require('morgan');
5 var cookieParser = require('cookie-parser');
6 var bodyParser = require('body-parser');
7
8 var routes = require('./routes/index');
9 var users = require('./routes/users');
10 var donations = require('./routes/donations.js');
11
12 var app = express();
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28 //Our Custom Routes
29 app.get('/donations', donations.findAll);
30 app.get('/donations/:id', donations.findOne);
31 app.post('/donations', donations.addDonation);
32 app.put('/donations/:id/votes', donations.incrementUpvotes);
33 app.delete('/donations/:id', donations.deleteDonation);
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68 module.exports = app;
69
70
71
```

# License

---

This course (slides, examples, demos, videos, homework, etc.) is licensed under the "[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International](#)" license



Attribution: this work may contain portions from

- "[Databases](#)" course by Telerik Academy under [CC-BY-NC-SA](#) license and



# 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?