NODE PROGRAM

MONGODB

NODE.JS VERSION: 5.1
LAST UPDATED: JAN 2016

C.A.P. THEOREM

A. CONSISTENCY (STRONG VS. EVENTUAL-DELAY)
B. AVAILABILITY
C. PARTITION TOLERANCE (ON CLUSTER)

NO SQL!

- A+P FROM C.A.P.
- NO RELATIONSHIPS IN THE DATABASE.
- REDUNDANCY IS GOOD.

NOSQL DATABASES

THERE ARE MANY TYPES OF NOSQL DATABASES:

- KEY-VALUE STORES (REDIS, THINK HASH TABLES)
- DOCUMENT STORES (MONGODB, THINK JSON)
- COLUMNAR STORES (HBASE, AVERAGE AGE)
- GRAPHS STORES (NEO4J)

SQL VS. NOSQL

RELATION DB->NORMILIZED FOR ANY QUERY, NO BIASES

NOSQL->BIASES TO SPECIFIC QUERY PATTERNS THAT WE HAVE

MONGODB

MONGODB IS A DOCUMENT STORE NOSQL DATABASE. IT'S GREAT AT DISTRIBUTED AND SCALING.

LAUNCHING MONGODB

LAUNCH THE `mongod` SERVICE WITH:

```
$ mongod
```

YOU SHOULD BE ABLE TO SEE INFORMATION IN YOUR TERMINAL. THE DEFAULT PORT IS 27017.

MONGODB SHELL (MONGO)

FOR THE MONGODB SHELL, OR MONGO, LAUNCH IN A NEW TERMINAL WINDOW (LET THE SERVER RUN), THIS COMMAND:

```
$ mongo
```

MONGODB SHELL (MONGO)

TO TEST THE DATABASE, USE THE JAVASCRIPT-LIKE INTERFACE AND COMMANDS SAVE AND FIND:

```
> db.test.save({a:1})
> db.test.find()
```

MONGODB USES JAVASCRIPT!

MONGODB SHELL (MONGO)

USEFUL MONGODB SHELL COMMANDS:

- `> help`
- `> show dbs`
- `> use board`
- `> show collections`
- `> db.messages.remove();`

MONGODB SHELL (MONGO)

USEFUL MONGODB SHELL COMMANDS:

- `> var a=db.messages.findOne();`
- `> print json(a);`
- `> a.message="hi";`
- `> db.messages.save(a);`
- `> db.messages.find({});`

MONGODB SHELL (MONGO)

USEFUL MONGODB SHELL COMMANDS:

- `> db.messages.update({name:"John"},{$set:{message:"bye"}});`
- `> db.messages.find({name:"John"});`
- `> db.messages.remove({name:"John"});`

DEMO

MONGODB NATIVE DRIVER VS. MONGODB SHELL

MONGODB NATIVE DRIVER (MONGODB)

NODE.JS NATIVE DRIVER FOR MONGODB (HTTPS://GITHUB.COM/CHRISTKV/NODE-MONGODB-NATIVE)

```
$ npm install mongodb --save
```

ESTABLISHING CONNECTION

```
var MongoClient = require('mongodb').MongoClient
  , assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server");

  db.close();
});
```

CREATING INSERTDOCUMENTS

```
var insertDocuments = function(db, callback) {
  // Get the documents collection
  var collection = db.collection('documents');
  // Insert some documents
  collection.insert([
    {a : 1}, {a : 2}, {a : 3}
  ], function(err, result) {
    assert.equal(err, null);
    assert.equal(3, result.result.n);
    assert.equal(3, result.ops.length);
    console.log("Inserted 3 documents into the document collection");
    callback(result);
  });
}
```

APPLYING INTERTDOCUMENTS

```
var MongoClient = require('mongodb').MongoClient
  , assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server");

  insertDocuments(db, function() {
    db.close();
  });
});
```

UPDATING DOCUMENTS

```
var updateDocument = function(db, callback) {
  // Get the documents collection
  var collection = db.collection('documents');
  // Update document where a is 2, set b equal to 1
  collection.update({ a : 2 }
    , { $set: { b : 1 } }, function(err, result) {
    assert.equal(err, null);
    assert.equal(1, result.result.n);
    console.log("Updated the document with the field a equal to 2");
    callback(result);
  });
}
```

APPLYING UPDATEDOCUMENT

```
insertDocuments(db, function() {
  updateDocument(db, function() {
    db.close();
  });
});
```

REMOVING DOCUMENTS

```
var removeDocument = function(db, callback) {
  // Get the documents collection
  var collection = db.collection('documents');
  // Insert some documents
  collection.remove({ a : 3 }, function(err, result) {
    assert.equal(err, null);
    assert.equal(1, result.result.n);
    console.log("Removed the document with the field a equal to 3");
    callback(result);
  });
}
```

APPLYING REMOVEDOCUMENT

```
var MongoClient = require('mongodb').MongoClient
  , assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server");

  insertDocuments(db, function() {
    updateDocument(db, function() {
      removeDocument(db, function() {
        db.close();
      });
    });
  });
});
```

FINDING DOCUMENTS

```
var findDocuments = function(db, callback) {
  // Get the documents collection
  var collection = db.collection('documents');
  // Find some documents
  collection.find({}).toArray(function(err, docs) {
    assert.equal(err, null);
    assert.equal(2, docs.length);
    console.log("Found the following records");
    console.dir(docs);
    callback(docs);
  });
}
```

APPLYING FINDDOCUMENTS

```
var MongoClient = require('mongodb').MongoClient
  , assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server");

  insertDocuments(db, function() {
    updateDocument(db, function() {
      removeDocument(db, function() {
        findDocuments(db, function() {
          db.close();
        });
      });
    });
  });
})
```

NATIVE DRIVER ALTERNATIVES

ALTERNATIVELY, FOR YOUR OWN DEVELOPMENT YOU COULD USE OTHER MAPPERS, WHICH ARE AVAILABLE AS AN EXTENSION OF THE NATIVE DRIVER:

- MONGOSKIN: THE FUTURE LAYER FOR NODE-MONGODB-NATIVE
- MONGOOSE: ASYNCHRONOUS JAVASCRIPT DRIVER WITH OPTIONAL SUPPORT FOR MODELING
- MONGOLIA: LIGHTWEIGHT MONGODB ORM/DRIVER WRAPPER
- MONK: MONK IS A TINY LAYER THAT PROVIDES SIMPLE YET

MONGODB BSON DATA TYPES

BINARY JSON, OR BSON, IT IS A SPECIAL DATA TYPE WHICH MONGODB UTILIZES. IT IS LIKE TO JSON IN NOTATION, BUT HAS SUPPORT FOR ADDITIONAL MORE SOPHISTICATED DATA TYPES.

[HTTP://BSONSPEC.ORG](http://bsonspec.org)

BINARY: THE BASE64 REPRESENTATION OF A BINARY STRING
DATE: A 64-BIT INTEGER OF THE ISO-8601 DATE FORMAT WITH A MANDATORY TIME ZONE FIELD FOLLOWING THE TEMPLATE YYYY-

MONGODB BSON DATA TYPES

TIMESTAMP: A 64 BIT VALUE
OID: A 24-CHARACTER HEXADECIMAL STRING
DB REFERENCE
MINKEY
MAXKEY
NUMBERLONG: A 64 BIT SIGNED INTEGER

QUESTIONS AND EXERCISES

❓🙋👍

WORKSHOP

🔨

```
$ [sudo] npm install -g learnyoumongo
```