Why MongoDB Is Awesome

DevNation Chicago, IL
May 15, 2010

John Nunemaker
Ordered List

I am user #4,243 on Twitter

@jnunemaker

“

…the best features of key/values stores, document databases and relational databases in one.

John Nunemaker

RailsTips.org June '09

Created by

10gen

I

♥

mongoDB

I

♥ A LOT

mongoDB

Which has led some people to believe

that I am on the payroll.

mongoDB
mongoDB
mongoDB
mongoDB
mongoDB
mongoDB
powered by
mongoDB
powered by

...but I am not.

I am merely a

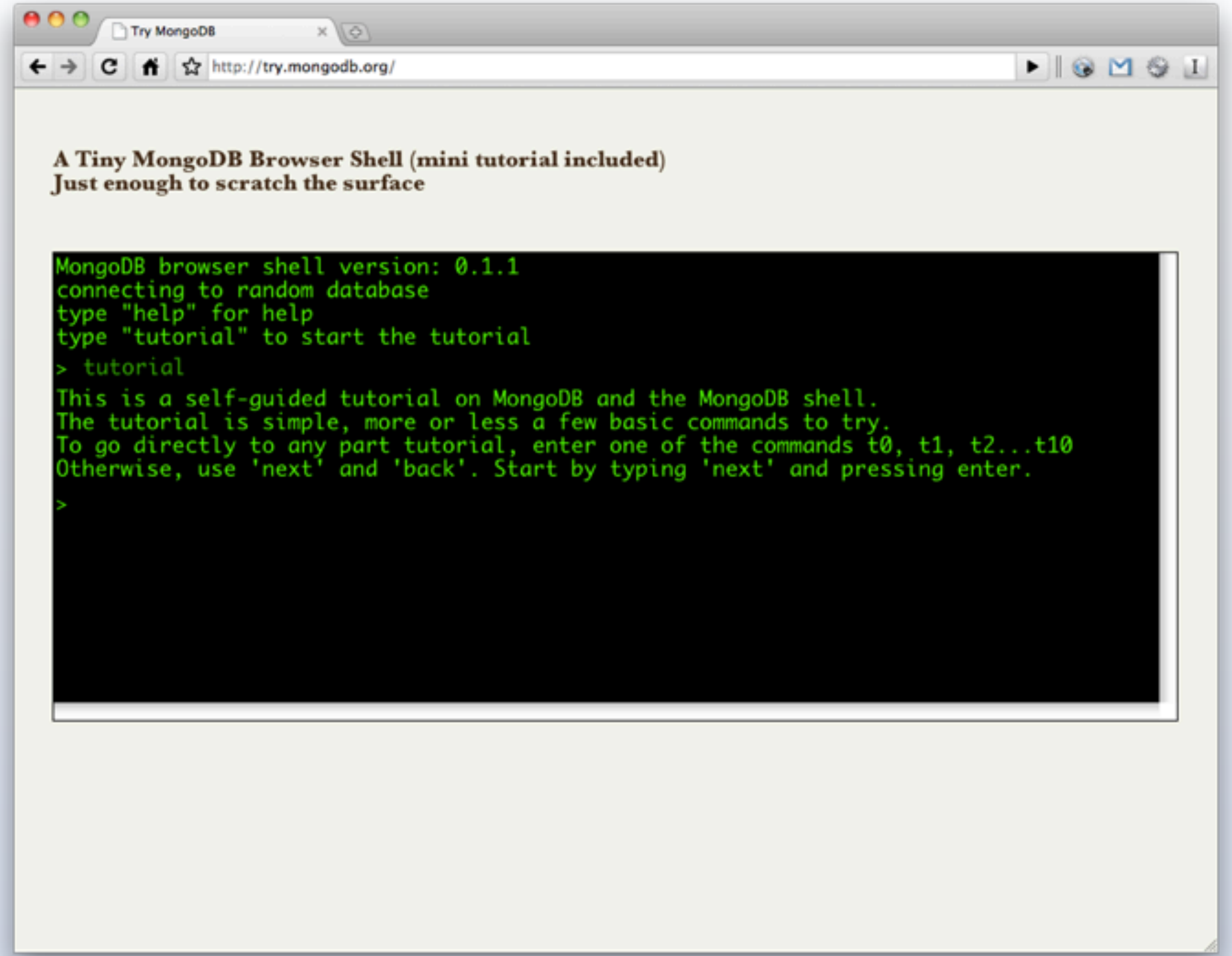
Satisfied User

Easy To

Try

Easy to Try

In Your Browser



A Tiny MongoDB Browser Shell (mini tutorial included)
Just enough to scratch the surface

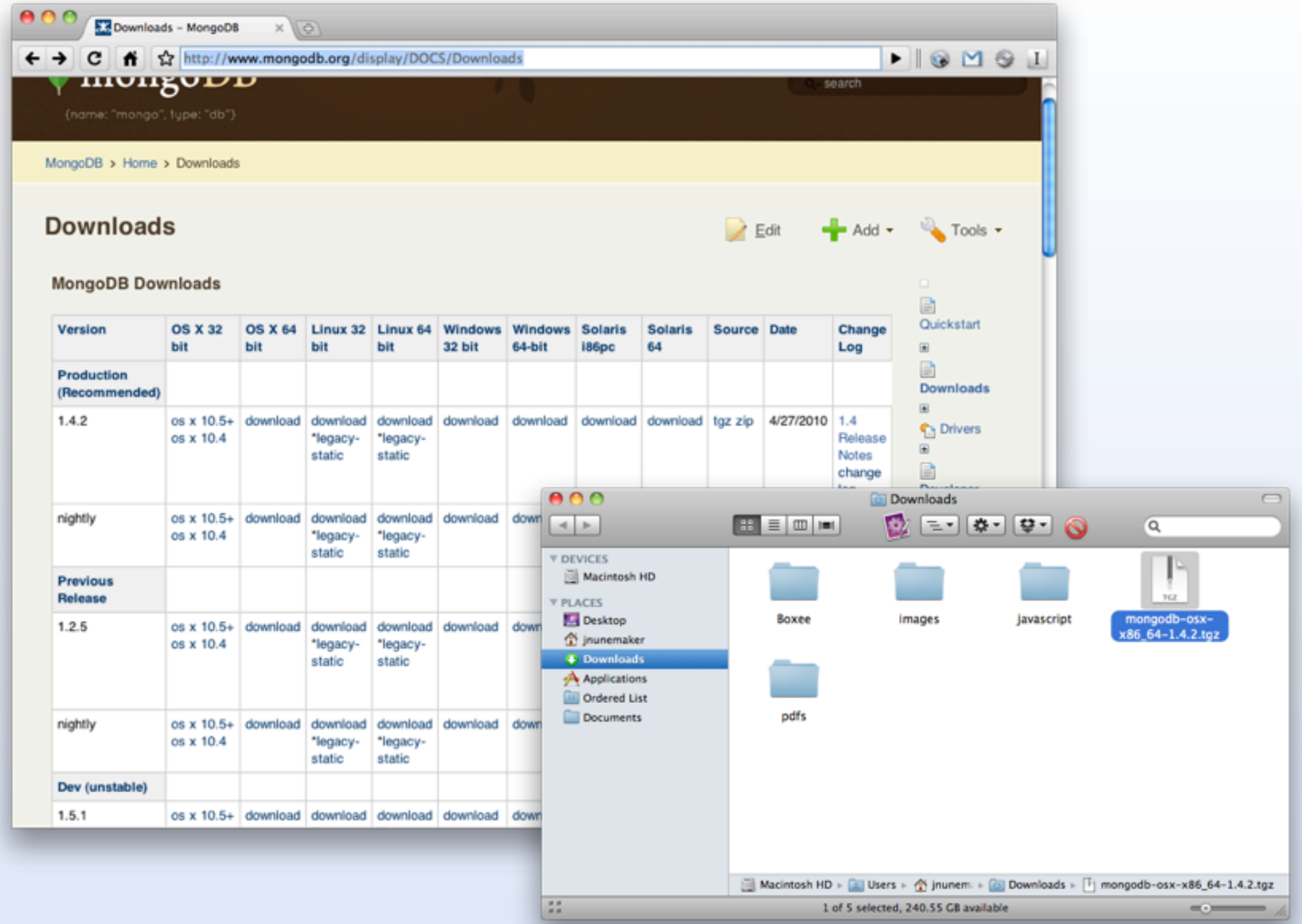
```
MongoDB browser shell version: 0.1.1
connecting to random database
type "help" for help
type "tutorial" to start the tutorial
> tutorial
This is a self-guided tutorial on MongoDB and the MongoDB shell.
The tutorial is simple, more or less a few basic commands to try.
To go directly to any part tutorial, enter one of the commands t0, t1, t2...t10
Otherwise, use 'next' and 'back'. Start by typing 'next' and pressing enter.
>
```

http://try.mongodb.org/

Easy to Try

On Your Computer

```
$ wget http://downloads.mongodb.org/osx/mongodb-osx-x86_64-1.4.2.tgz
$ wget http://downloads.mongodb.org/osx/mongodb-osx-x86_64-1.4.2.tgz
$ tar -xf mongodb-osx-x86_64-1.4.2.tgz
$ wget http://downloads.mongodb.org/osx/mongodb-osx-x86_64-1.4.2.tgz
$ tar -xf mongodb-osx-x86_64-1.4.2.tgz
$ mkdir -p /data/db
$ wget http://downloads.mongodb.org/osx/mongodb-osx-x86_64-1.4.2.tgz
$ tar -xf mongodb-osx-x86_64-1.4.2.tgz
$ mkdir -p /data/db
$ mongodb-osx-x86_64-1.4.2/bin/mongod
```


Downloads - MongoDB
http://www.mongodb.org/display/DOCS/Downloads
{name: "mongo", type: "db"}
search
MongoDB > Home > Downloads
Edit
Add
Tools
Quickstart
Downloads
Drivers

Downloads

MongoDB Downloads

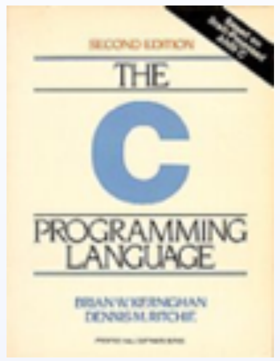
Version	OS X 32 bit	OS X 64 bit	Linux 32 bit	Linux 64 bit	Windows 32 bit	Windows 64-bit	Solaris i86pc	Solaris 64	Source	Date	Change Log
Production (Recommended)											
1.4.2	os x 10.5+ os x 10.4	download	download *legacy-static	download *legacy-static	download	download	download	download	tgz zip	4/27/2010	1.4 Release Notes change
nightly	os x 10.5+ os x 10.4	download	download *legacy-static	download *legacy-static	download	down					
Previous Release											
1.2.5	os x 10.5+ os x 10.4	download	download *legacy-static	download *legacy-static	download	down					
nightly	os x 10.5+ os x 10.4	download	download *legacy-static	download *legacy-static	download	down					
Dev (unstable)											
1.5.1	os x 10.5+	download	download	download	download	down					

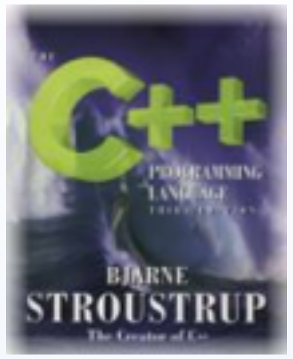
Downloads
DEVICES
Macintosh HD
PLACES
Desktop
jnunemaker
Downloads
Applications
Ordered List
Documents
Boxee
images
javascript
mongodb-osx-x86_64-1.4.2.tgz
pdfs
Macintosh HD ▸ Users ▸ jnunem ▸ Downloads ▸ mongodb-osx-x86_64-1.4.2.tgz
1 of 5 selected, 240.55 GB available

http://www.mongodb.org/display/DOCS/Downloads

Easy to Try

From Your Language





http://www.mongodb.org/display/DOCS/Drivers

Easy To

Understand

Easy to Understand

Similar Terms

Database == Database

```
> show dbs
    admin
    harmony-development
    harmony-test
    local
    ...
> use harmony-development
    switched to db harmony-development
> show collections
    accounts
    activities
    assets
    items
    ...
```

Collection == Table

```
> db.accounts
harmony-development.accounts

> db.accounts.count()
1

> db.accounts.find().forEach(function(doc) {
  print(tojson(doc));
});
```

Document == Row

```
{
  "_id"         : ObjectId("4be97eaebcd1b30e86000003"),
  "title"       : "Ordered List",
  "creator_id"  : ObjectId("4be97eadbcd1b30e86000001"),
  "memberships" : [
    ObjectId("4be97eadbcd1b30e86000001"),
    ObjectId("4be97eaebcd1b30e86000002")
  ]
}
```

Easy to Understand

Similar Functionality

Dynamic Queries

http://www.mongodb.org/display/DOCS/Querying

http://www.mongodb.org/display/DOCS/Advanced+Queries

```
> use testing
switched to db testing

> db.colors.insert({name:'red',    primary:true})
> db.colors.insert({name:'green',  primary:true})
> db.colors.insert({name:'blue',   primary:true})
> db.colors.insert({name:'purple', primary:false})
> db.colors.insert({name:'orange', primary:false})
> db.colors.insert({name:'yellow', primary:false})
> var cursor = db.colors.find()
> cursor.next()
{
  "_id" : ObjectId("4bed7aeb0b4acd070c593ba6"),
  "name" : "red",
  "primary" : true
}
```

```
> cursor
{ "_id" : ObjectId("4bed7af40b4acd070c593ba7"), "name" : "green", "primary" : true }
{ "_id" : ObjectId("4bed7af80b4acd070c593ba8"), "name" : "blue", "primary" : true }
{ "_id" : ObjectId("4bed7b570b4acd070c593ba9"), "name" : "purple", "primary" : false }
{ "_id" : ObjectId("4bed7b6a0b4acd070c593baa"), "name" : "orange", "primary" : false }
{ "_id" : ObjectId("4bed7b7d0b4acd070c593bab"), "name" : "yellow", "primary" : false }
```

```
SELECT * from colors WHERE name = 'green'
SELECT * from colors WHERE name = 'green'
```

```
> db.colors.find({name:'green'})
{ "_id" : ObjectId("4bed7af40b4acd070c593ba7"), "name" : "green", "primary" : true }
```

```sql
SELECT name from colors WHERE primary = 1
SELECT name from colors WHERE primary = 1
```

```
> db.colors.find({primary:true}, {name:true})
{ "_id" : ObjectId("4bed7aeb0b4acd070c593ba6"), "name" : "red" }
{ "_id" : ObjectId("4bed7af40b4acd070c593ba7"), "name" : "green" }
{ "_id" : ObjectId("4bed7af80b4acd070c593ba8"), "name" : "blue" }
```

```
> db.colors.find({name:/l/})
{ "_id" : ObjectId("4bed7af80b4acd070c593ba8"), "name" : "blue", "primary" : true }
{ "_id" : ObjectId("4bed7b570b4acd070c593ba9"), "name" : "purple", "primary" : false }
{ "_id" : ObjectId("4bed7b7d0b4acd070c593bab"), "name" : "yellow", "primary" : false }
```

```
> db.colors.find({primary:true}).sort({name:1}).limit(1)
{ "_id" : ObjectId("4bed7af80b4acd070c593ba8"), "name" : "blue", "primary" : true }
```

```
> db.colors.find({primary:true}).sort({name:1}).limit(1)
{ "_id" : ObjectId("4bed7af80b4acd070c593ba8"), "name" : "blue", "primary" : true }
```

```
> db.colors.find({primary:true}).sort({name:-1}).limit(1)
{ "_id" : ObjectId("4bed7aeb0b4acd070c593ba6"), "name" : "red", "primary" : true }
```

```
> db.colors.find({primary:true}).sort({name:1}).limit(1)
{ "_id" : ObjectId("4bed7af80b4acd070c593ba8"), "name" : "blue", "primary" : true }
```

```
> db.colors.find({primary:true}).sort({name:-1}).limit(1)
{ "_id" : ObjectId("4bed7aeb0b4acd070c593ba6"), "name" : "red", "primary" : true }
```

```
> db.colors.find({primary:true}).sort({name:1}).skip(1).limit(1)
{ "_id" : ObjectId("4bed7af40b4acd070c593ba7"), "name" : "green", "primary" : true }
```

```
> db.people.insert({name:'John',  age:28})
> db.people.insert({name:'Steve', age:29})
> db.people.insert({name:'Steph', age:27})
```

```
SELECT * from people WHERE age > 27
SELECT * from people WHERE age > 27
```

```
> db.people.find({age: {$gt: 27}})
{ "_id" : ObjectId("4bed80b20b4acd070c593bac"), "name" : "John", "age" : 28 }
{ "_id" : ObjectId("4bed80bb0b4acd070c593bad"), "name" : "Steve", "age" : 29 }
```

```
SELECT * from people WHERE age <= 27
SELECT * from people WHERE age <= 27

> db.people.find({age: {$lte: 27}})
{ "_id" : ObjectId("4bed80c10b4acd070c593bae"), "name" : "Steph", "age" : 27 }
```

```
$gt         $all
$gte        $size
$lt         $exists
$lte        $type
$ne         $elemMatch
$in         $not
$nin        $where
$mod
```

Indexes

http://www.mongodb.org/display/DOCS/Indexes

```
// single ascending
> db.colors.ensureIndex({name: 1})
```

```
// single ascending
> db.colors.ensureIndex({name: 1})

// single descending
> db.colors.ensureIndex({created_at: -1})
```

```
// single ascending
> db.colors.ensureIndex({name: 1})

// single descending
> db.colors.ensureIndex({created_at: -1})

// unique
> db.colors.ensureIndex({email: 1}, {unique: true})
```

```
// single ascending
> db.colors.ensureIndex({name: 1})

// single descending
> db.colors.ensureIndex({created_at: -1})

// unique
> db.colors.ensureIndex({email: 1}, {unique: true})

// non-blocking in background
> db.colors.ensureIndex({name: 1}, {background: true})
```

```
// single ascending
> db.colors.ensureIndex({name: 1})

// single descending
> db.colors.ensureIndex({created_at: -1})

// unique
> db.colors.ensureIndex({email: 1}, {unique: true})

// non-blocking in background
> db.colors.ensureIndex({name: 1}, {background: true})

// compound
> db.colors.ensureIndex({name: 1, created_at: -1})
```

Aggregation

http://www.mongodb.org/display/DOCS/Aggregation

```
> db.colors.count()
6
> db.colors.count
({primary:true})
3
```

```
> db.colors.distinct('name')
[ "blue", "green", "orange", "purple", "red", "yellow" ]


> db.people.distinct('name', {age:28})
[ "John" ]
```

```
> db.items.insert({title:'Home',         template:'home'})
> db.items.insert({title:'What We Do',   template:'page'})
> db.items.insert({title:'Our Writing', template:'page'})
> db.items.insert({title:'Who We Are',   template:'page'})
> db.items.insert({title:'Hire Us',      template:'page'})



> var key     = {template: true};
> var initial = {count:0};
> var reduce  = function(obj, prev) { prev.count += 1; };

> db.items.group({key:key, initial:initial, reduce:reduce})
[
  {"template" : "home", "count" : 1},
  {"template" : "page", "count" : 4}
]
```

```
> db.items.insert({tags: ['dog', 'cat']})
> db.items.insert({tags: ['dog']})
> db.items.insert({tags: ['dog', 'mouse']})
> db.items.insert({tags: ['dog', 'mouse', 'hippo']})
> db.items.insert({tags: ['dog', 'mouse', 'hippo']})
> db.items.insert({tags: ['dog', 'hippo']})
> var map = function() {
    this.tags.forEach(function(t) {
      emit(t, {count: 1});
    });
  }
```

```
> var reduce = function(key, values) {
    var count = 0;
    for(var i=0, len=values.length; i<len; i++) {
      count += values[i].count;
    }
    return {count: count};
  }
```

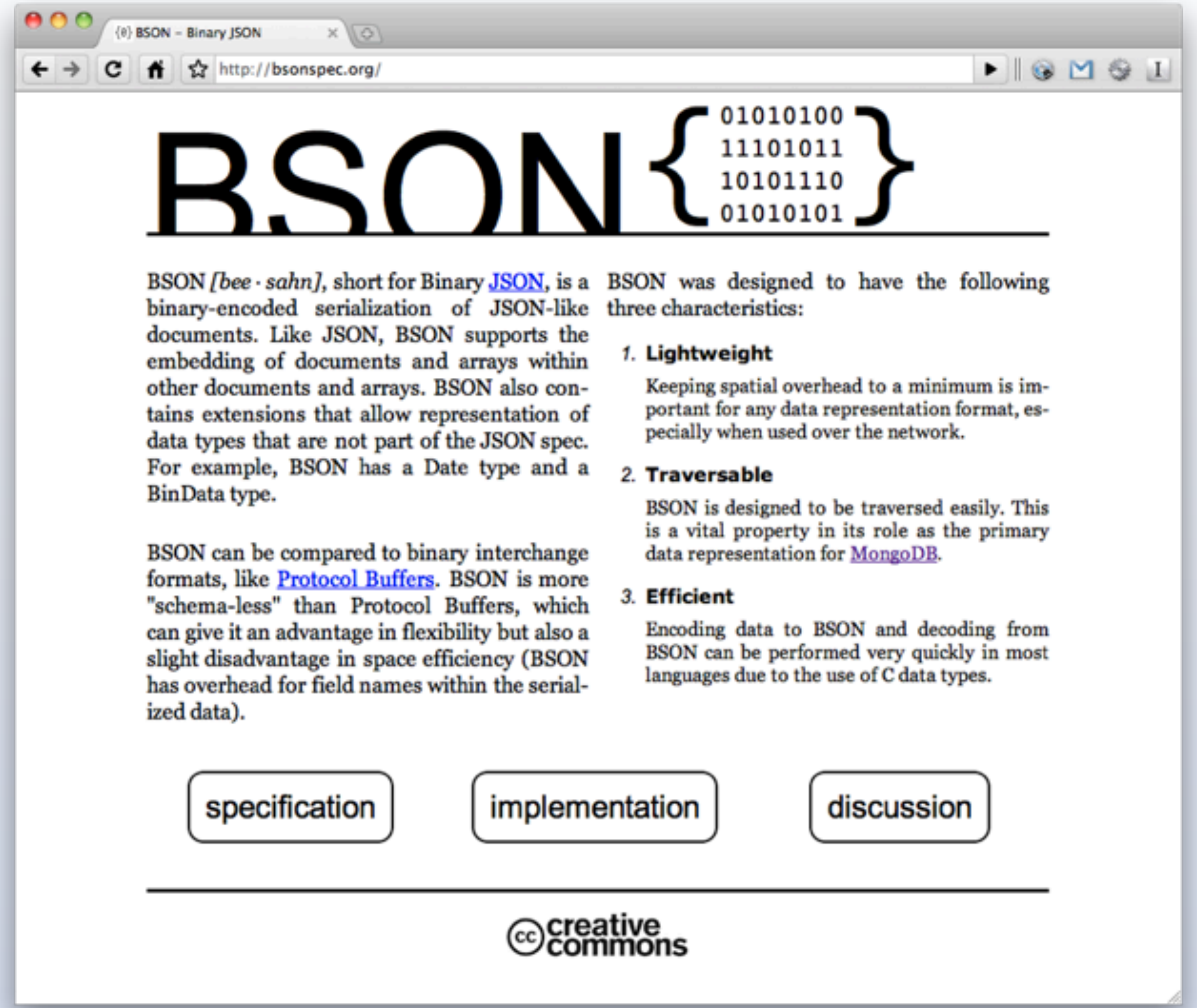
```
> var result = db.items.mapReduce(map, reduce);
> var result = db.items.mapReduce(map, reduce);
> result
{
  "ok"          : 1,
  "timeMillis"  : 86,
  "result"      : "tmp.mr.mapreduce_1273861517_683",
  "counts"      : {
    "input"     : 6,
    "emit"      : 13,
    "output"    : 4
  }
}
```

```
> db[result.result].find()
{ "_id" : "cat",   "value" : { "count" : 1 } }
{ "_id" : "dog",   "value" : { "count" : 6 } }
{ "_id" : "hippo", "value" : { "count" : 3 } }
{ "_id" : "mouse", "value" : { "count" : 3 } }
```

Easy to Understand

Similar Data Types

Array, Binary, Boolean, DateTime, DB Reference, Embedded Object, Integer, Null, ObjectId, RegExp, String, Symbol, Timestamp



BSON { 01010100 11101011 10101110 01010101 }

BSON *[bee · sahn]*, short for Binary JSON, is a binary-encoded serialization of JSON-like documents. Like JSON, BSON supports the embedding of documents and arrays within other documents and arrays. BSON also contains extensions that allow representation of data types that are not part of the JSON spec. For example, BSON has a Date type and a BinData type.

BSON can be compared to binary interchange formats, like Protocol Buffers. BSON is more "schema-less" than Protocol Buffers, which can give it an advantage in flexibility but also a slight disadvantage in space efficiency (BSON has overhead for field names within the serialized data).

BSON was designed to have the following three characteristics:

1. **Lightweight**

   Keeping spatial overhead to a minimum is important for any data representation format, especially when used over the network.

2. **Traversable**

   BSON is designed to be traversed easily. This is a vital property in its role as the primary data representation for MongoDB.

3. **Efficient**

   Encoding data to BSON and decoding from BSON can be performed very quickly in most languages due to the use of C data types.

```
> db.people.insert({
  name    : 'John',
  awesome : true,
  shows   : ['Dexter', 'LOST', 'How I Met Your Mother'],
  info    : {
    age : 28,
    home: 'South Bend, IN',
    dob : (new Date('November 25, 1981'))
  }
})
```

```
> var me = db.people.findOne({name:'John'})
> me.name
John
> me.awesome
true
> me.shows[1]
LOST
> me.info.age
28
> me.info.dob.getFullYear()
1981
```

```
> db.people.find({'info.age': 28})
{ "_id" : ObjectId("4bed9cba0b4acd070c593bc5"), "name" : "John" }
```

```
> db.people.find({'info.age': 28})
{ "_id" : ObjectId("4bed9cba0b4acd070c593bc5"), "name" : "John" }
```

```
> db.people.find({shows:'Dexter'})
{ "_id" : ObjectId("4bed9cba0b4acd070c593bc5"), "name" : "John" }
```

```
> db.people.find({'info.age': 28})
{ "_id" : ObjectId("4bed9cba0b4acd070c593bc5"), "name" : "John" }
```

```
> db.people.find({shows:'Dexter'})
{ "_id" : ObjectId("4bed9cba0b4acd070c593bc5"), "name" : "John" }
```

```
> db.people.find({shows:{$in:['Dexter', 'LOST']}})
{ "_id" : ObjectId("4bed9cba0b4acd070c593bc5"), "name" : "John" }
```

Easy to Understand

Similar Relationships

One to Many

1. Normalized

```
// insert post
> db.posts.insert({title:'Why Mongo Rocks'});
> var post = db.posts.findOne({title:'Why Mongo Rocks'});
```

```
// insert post
> db.posts.insert({title:'Why Mongo Rocks'});
> var post = db.posts.findOne({title:'Why Mongo Rocks'});


// insert comment
> db.comments.insert({
  name    :'John',
  body    :'Because...',
  post_id : post._id
});
> var comment = db.comments.findOne({name:'John'});
```

```
SELECT * FROM comments WHERE post_id = #{post.id}
```

```
> db.comments.find({post_id: post._id})
{
  "_id"      : ObjectId("4bee1cc79e89db4e12bf78de"),
  "name"     : "John",
  "body"     : "Because...",
  "post_id"  : ObjectId("4bee1c519e89db4e12bf78dd")
}
```

```
SELECT * FROM posts WHERE id = #{comment.id}
```

```
> db.posts.find({_id: comment.post_id})
{
  "_id"   : ObjectId("4bee1c519e89db4e12bf78dd"),
  "title" : "Why Mongo Rocks"
}
```

2. Embedded

```
// insert post AND comments
> db.posts.insert({
  title:'Why Mongo Rocks',
  comments: [
    {name:'John', body:'Because...'},
    {name:'Steve', body:'Uh huh!'}
  ]
})
```

```
> var post = db.posts.find({title:'Why Mongo Rocks'});
> var post = db.posts.find({title:'Why Mongo Rocks'});
> post
{
  "_id"      : ObjectId("4bee21259e89db4e12bf78df"),
  "title"    : "Why Mongo Rocks",
  "comments" : [
    {"name": "John", "body": "Because..."},
    {"name": "Steve", "body": "Uh huh!"}
  ]
}
```

```
> db.posts.find({'comments.name':'John'})
> db.posts.find({'comments.name':'John'})

> db.posts.find({
  comments: {
    $elemMatch: {name:'John'}
  }
})
```

```
// insert post AND comments AND threads!
> db.posts.insert({
  title:'Why Mongo Rocks',
  comments: [
    {
      name:'John',
      body:'Because...',
      comments: [
        {name:'Frank', body:'You are crazy!'},
        {name:'Billy', body:'Frank Furter!'}
      ]
    }
  ]
})
```

```
> db.posts.insert({
  title : 'Why Mongo Rocks',
  tags  : ['mongodb', 'databases']
})
```

```
> db.posts.insert({
  title : 'Why Mongo Rocks',
  tags  : ['mongodb', 'databases']
})
> db.posts.ensureIndex({tags:1})
```

Some Notes

Some Notes

- Embedding is pre-joining

Some Notes

- Embedding is pre-joining
- Embed when document always appears with parent

Some Notes

- Embedding is pre-joining
- Embed when document always appears with parent
- 4MB document size limit

Many to Many

```
> db.sites.insert({domain: 'orderedlist.com'})
> db.sites.insert({domain: 'railstips.org'})
> db.sites.find()
{
  "_id"   : ObjectId("4bee280f9e89db4e12bf78e2"),
  "domain": "orderedlist.com"
}
{
  "_id"   : ObjectId("4bee283c9e89db4e12bf78e3"),
  "domain": "railstips.org"
}
```

```
> db.users.insert({
  name: 'John',
  authorizations: [
    ObjectId('4bee280f9e89db4e12bf78e2'),
    ObjectId('4bee283c9e89db4e12bf78e3')
  ]
})


> db.users.insert({
  name: 'Steve',
  authorizations: [
    ObjectId('4bee280f9e89db4e12bf78e2')
  ]
})
```

```
> var orderedlist = db.sites.findOne({domain:'orderedlist.com'})
> db.users.find({authorizations:orderedlist._id})
// john and steve
```

```
> var railstips = db.sites.findOne({domain:'railstips.org'})
> db.users.find({authorizations:railstips._id})
// john
```

```
> var john = db.users.findOne({name:'John'})
> db.sites.find({_id:{$in: john.authorizations}})
// orderedlist.com and railstips.org
```

Easy To

Learn

By Email

http://groups.google.com/group/mongodb-user

By IRC

irc://irc.freenode.net/#mongodb

By Web

http://mongodb.org/

http://mongotips.com/

By Book

http://www.10gen.com/books

http://cookbook.mongodb.org/

By Conference

http://www.10gen.com/events

http://windycitydb.org/

By Training

http://ideafoundry.info/mongodb

Thank you!

john@orderedlist.com

@jnunemaker

DevNation Chicago, IL
May 15, 2010

John Nunemaker
Ordered List