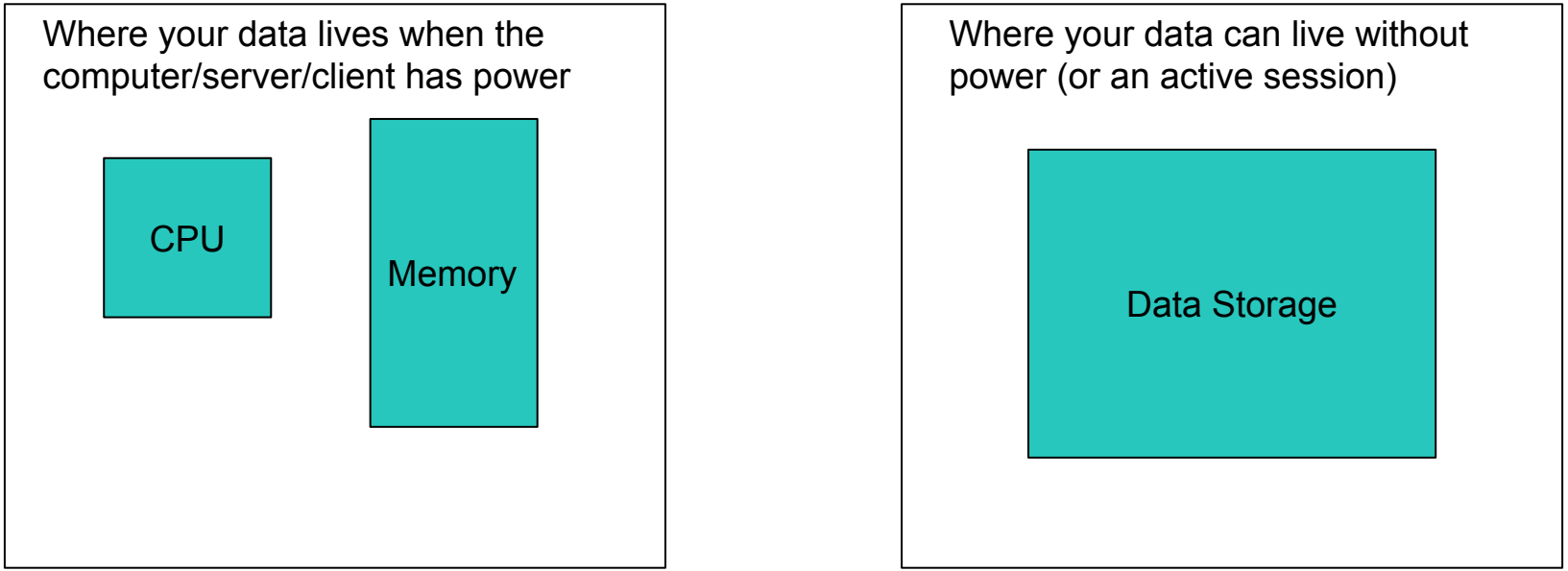
Meet MongoDB

Storing your data forever ever

On Data Storage



Why data storage (high-level)

3 ways to store data

- In files on the file system
- In a relational database
- In a NoSQL database

Why not the file system?

- Querying data from a file is slow and painful
- Databases are made for efficiently storing and retrieving data

MongoDB

Highlights

- Document-oriented
- High performance
- High availability
- High scalability

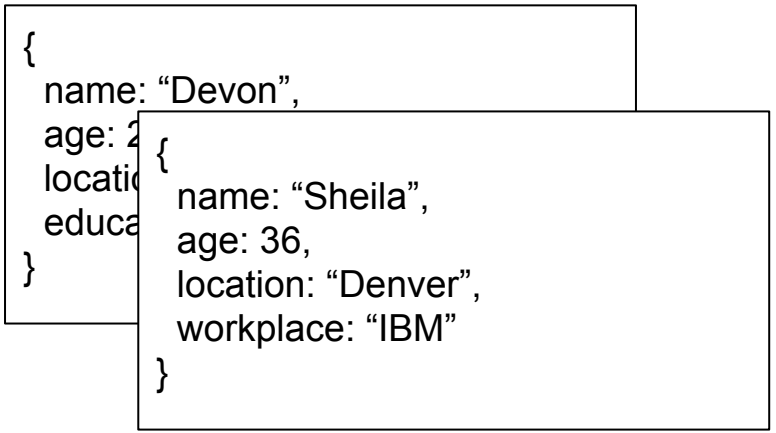
Documents

A single entity of data

```
{
  name: “Devon”,
  age: 25,
  location: “Chicago”,
  education: [“Kenwood HS”, “IIT”]
}
```

Collections

A grouping of documents with the same purpose



Querying MongoDB

Why is this happening?

- Become familiar with MongoDB syntax
- Become familiar working with MongoDB outside of the scope of a Node app

MongoDB Shell

The MongoDB shell

We can interact with our database via the terminal by running MongoDB's shell program: an interactive JavaScript-based shell.

Accessing the MongoDB shell

1. Go to your terminal
2. Type `mongod` to start MongoDB
3. Open a new terminal
4. Type `mongo` to start MongoDB shell

Mongo shell commands

Typing `help` shows you a variety of commands you can use in the shell.

MongoDB Shell Objects

Objects

1. Connection
2. Database
3. Collection

Database object

- Provides access to databases
- Acts as a representation of the database upon which you can call methods

```
> use <database> // accesses a specific database or
                 // creates it
> db.getName();  // outputs name of database
                 // where db represents the current
                 // database
```

Collection object

- Provides access to collections
- Use it to add and query documents

```
> db.createCollection(“myCollection”)
// create a new collection in the current database
> db.myCollection // use the dot notation to access a
                  // collection in the current db
```

Query Time (Basic)

Rando is back!

Rando, the HR/Talent company, has decided to store its talent and company data in a NoSQL database. They've chosen MongoDB, just our luck!

Let's build them a database!

The queries

- Create a database
- Create a collection
- Add a document to a collection
- Section: Finding Documents

Create a database

```
> use rando
```

Create a collection

```
> db.createCollection(“talent”)
```

Add a document to a collection

{"talent_id":32, "name":"Michael Jordan", "address": "555 Chicago Lane, Chicago, IL 33224", "skills":["basketball", "entrepreneurship", "product creation"], "salary_req":[{"min": 200000, "max": 500000,"type": "annual"} , {"min": 250, "max": 400, "type": "hourly"}],"current": null}

{"talent_id":60, "name":"Aimee Mann", "address": "123 Main St, Portland, OR", "skills":["music", "acting"],"salary_req":[{"min": 100000, "max": 200000, "type": "annual"}, {"min": 100, "max": 500, "type": "hourly"}], "history":[{ "org":"Music Music", "start": new Date("Jan 04,2000"), "end": new Date("Nov 24, 2012")}]," current": "Portlandia"}

Aside: data types

- Date and Null are one of a few BSON data types.
- We can use data types for refining queries.

[MongoDB data types]

Add a document to a collection

```
> db.talent.insert(<doc or array of docs>)
```

Finding Documents: An Odyssey

Find a single document

```
> db.talent.findOne() // get a document
```

Find all documents

```
> db.talent.find() // returns a cursor object
                   // that represents the found
                   // records
> db.talent.find().pretty() // makes results
                            // more readable
```

Explanation: Cursor object

- Certain queries return a Cursor object
- Acts as a pointer that can be iterated on

```
> var result = db.talent.find()
> result.next() // display the next record
```

Find by field value

```
> db.talent.find({"current": null})
```

Find by array of values

```
> db.talent.find({"talent_id":{$in: [32, 60,
64]}})
```

Find by array of values

```
> db.talent.find({"talent_id":{$in: [32, 60,
64]}})
```

Find by less than, greater than

```
> db.talent.find({"talent_id":{$gt: 40}})
> db.talent.find({"talent_id":{$lt: 40}})
```

Find by array contents

```
> db.talent.find({"skills":{$all: ["basketball"]}})
```

Find by existence of field

```
> db.talent.find({"history":{$exists: true}})
```

Find based on fields in array of subdocuments

```
> db.talent.find({"salary_req":{$elemMatch:{"min":{$lt: 120000}}}}) // does not return what we really want
> db.talent.find({"salary_req":{$elemMatch:{$and:[{"min": {$lt: 120000}}, {"type":"annual"}]}}})}}) // add the $and operator
```

Etcetera

Rando talent data

- TalentID
- Name
- Address
- Skills
- Salary Requirements
- Employment History
- Current Placement

Resources

- Query Operators
- Cursor Iteration

Loved it? Hated it?

Provide your daily feedback right here: http://goo.gl/forms/drAE4Z9xAY