Cubstart Lecture 7

MongoDB and Mongoose

[start recording]

Administrivia

- HW 6: Quizlet-ish Part 1 due on Friday
- HW 7: Quizlet-ish Part 2 due next Friday
- Please check HW feedback and HW Ed posts!!!!
- LAB THIS WEEK IS AN OPTIONAL MAKE-UP LAB
- Team formation form DUE IN ONE WEEK
  - https://forms.gle/2yAcxgQ67EUCkiFo8
  - Let us know who you want to work with and your preferred team size :)

Feedback Form

https://forms.gle/YEJoNmV2UswEw7Fi7

Agenda:

- Homework 5: Common Mistakes
- Big Picture: APIs and Databases
- MongoDB
- Mongoose

Homework 5 Review

- Calling 2 OpenWeatherMap APIs
  - Geocoding: Get lat and lon from a city name
  - Current Weather: Get weather from lat and lon
- Script.js -> Send GET request to API -> API sends JSON back -> Parse JSON and display data on front-end

Homework 5 Common Mistakes

```
apiKey = "blablabla123fakeapikey";
geocodingUrl = "http://api.openweathermap.org/geo/1.0/direct?q=";
weatherUrl = "https://api.openweathermap.org/data/2.5/weather?";
```

API call

```
http://api.openweathermap.org/geo/1.0/direct?q={city name},
{state code},{country code}&limit={limit}&appid={API key}
```

Both of these are fine:

```
let url = geocodingUrl + city + "&appid=" + apiKey;
let url2 = "http://api.openweathermap.org/geo/1.0/direct?q=" + city + "&appid=" + apiKey;
```

Homework 5 Common Mistakes

API response body:


```
{
    "coord": {
        "lon": 5,
        "lat": 14
    },
    "weather": [
        {
            "id": 804,
            "main": "Clouds",
            "description": "overcast clouds",
            "icon": "04n"
        }
    ],
    "base": "stations",
```


Both of these are fine:

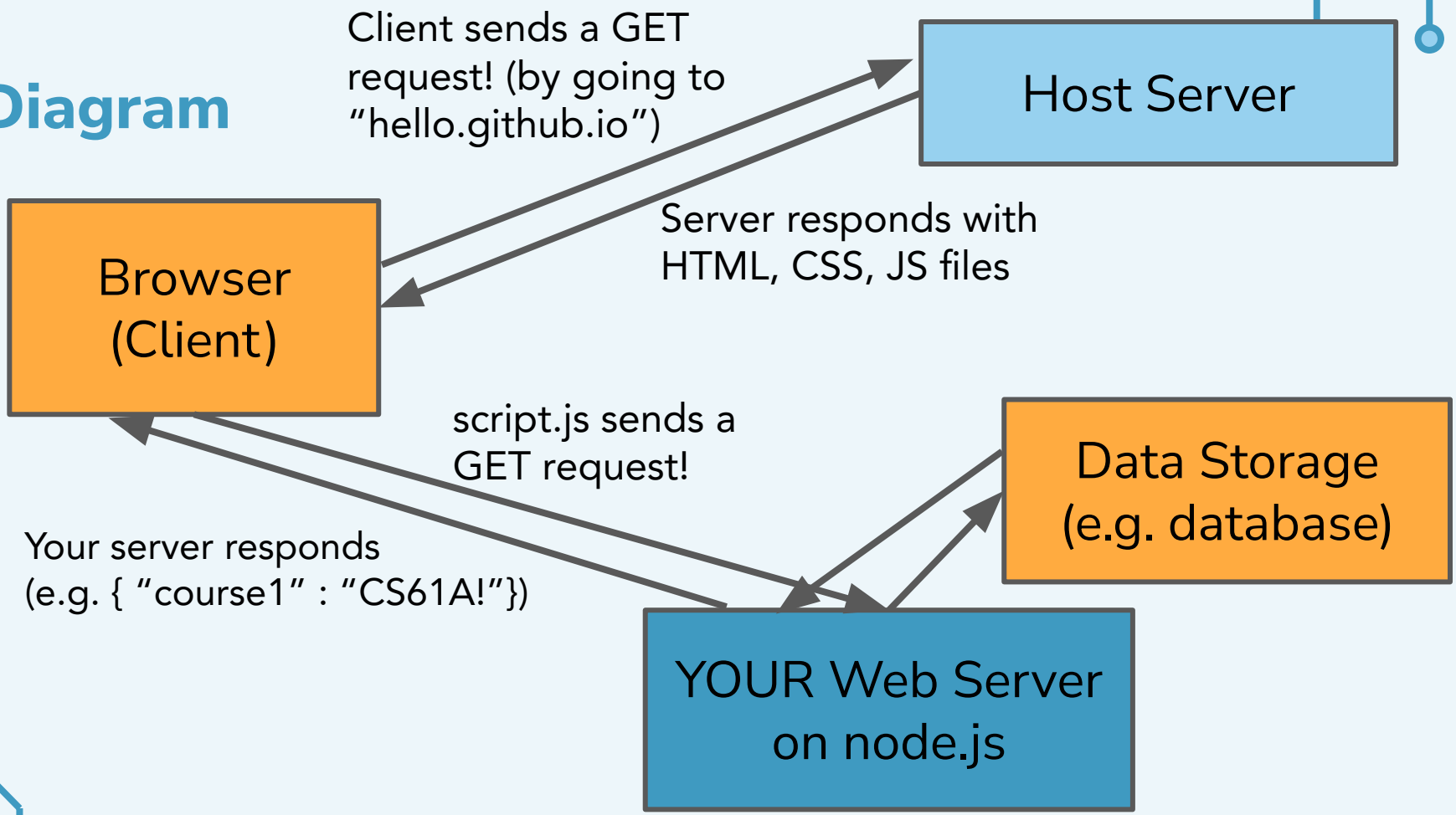
```
return {
  "main": data.weather[0].main,
  "description": data.weather[0].description
}
```

```
return {
  "main": data["weather"][0]["main"],
  "description": data["weather"][0]["description"]
}
```

Big Picture: APIs and Databases

- Every time you run “node index.js”, the server restarts, data does **not** persist
  - HW6: We store data in an array that “refreshes” every time we restart the server
- **Databases:** Allow us to **keep data!**

Diagram


Client sends a GET request! (by going to “hello.github.io”)
Host Server
Server responds with HTML, CSS, JS files
Browser (Client)
script.js sends a GET request!
Data Storage (e.g. database)
Your server responds (e.g. { “course1” : “CS61A!”})
YOUR Web Server on node.js

Big Picture: APIs and Databases

- Send request to API endpoints -> API server modifies database -> Database stores data
- If we reload server, data **remains in database!**

Pseudocode (index.js)

// connect to database!

// define endpoint “/new” (to add a new flashcard) for POST requests

    // save data from POST request body to database

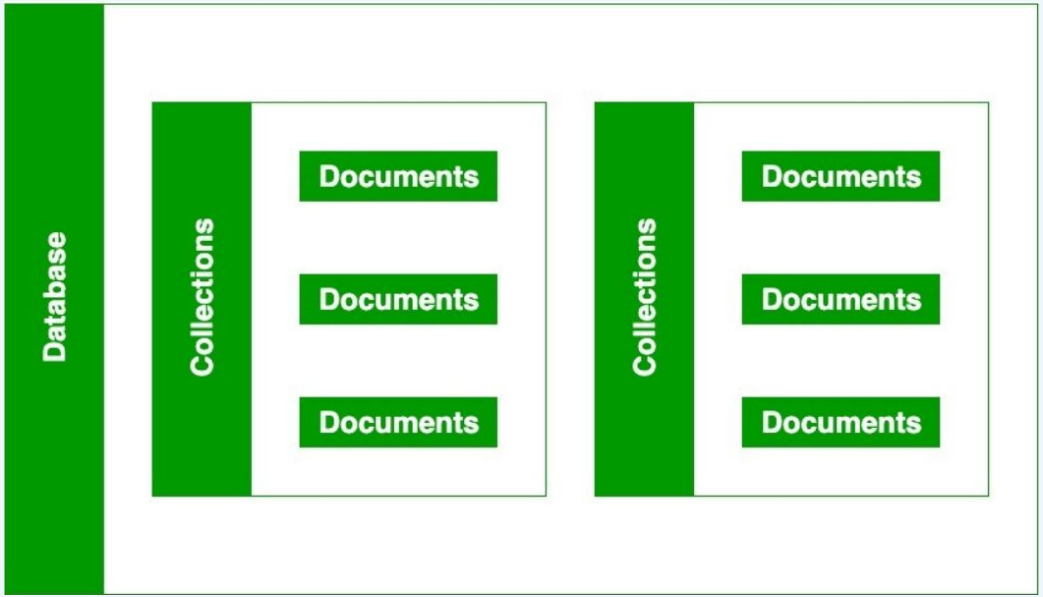
MongoDB

What is MongoDB?

mongoDB®

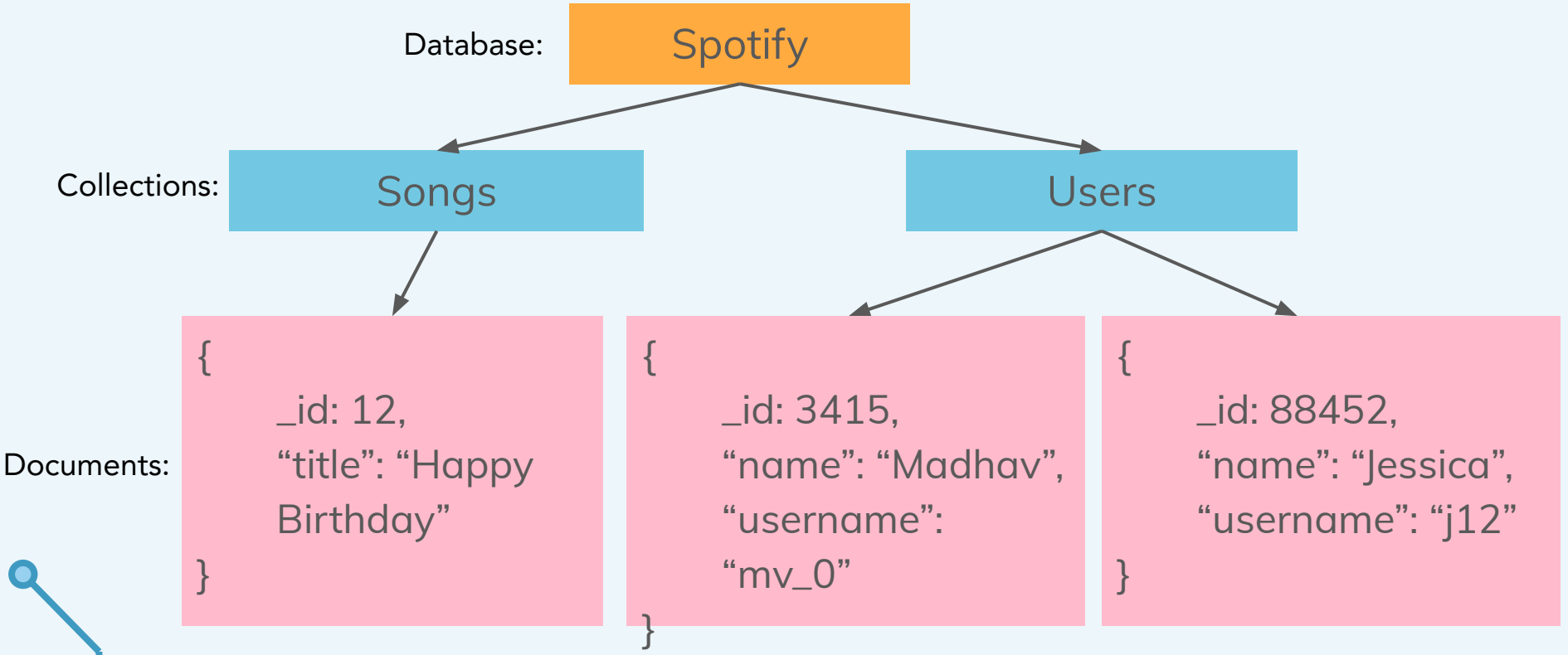
- “MongoDB is a document database used to build highly available and scalable internet applications.”
- NoSQL database: Instead of storing data in tables of rows or columns like SQL databases, each record in a MongoDB database is a document
- You can retrieve/store data in documents as JSON
- You can programmatically access, modify, and delete things in the database
- Driven by JavaScript!
- Well-supported and well-documented
- Used by Discord, Google, etc.

How is data stored in MongoDB?

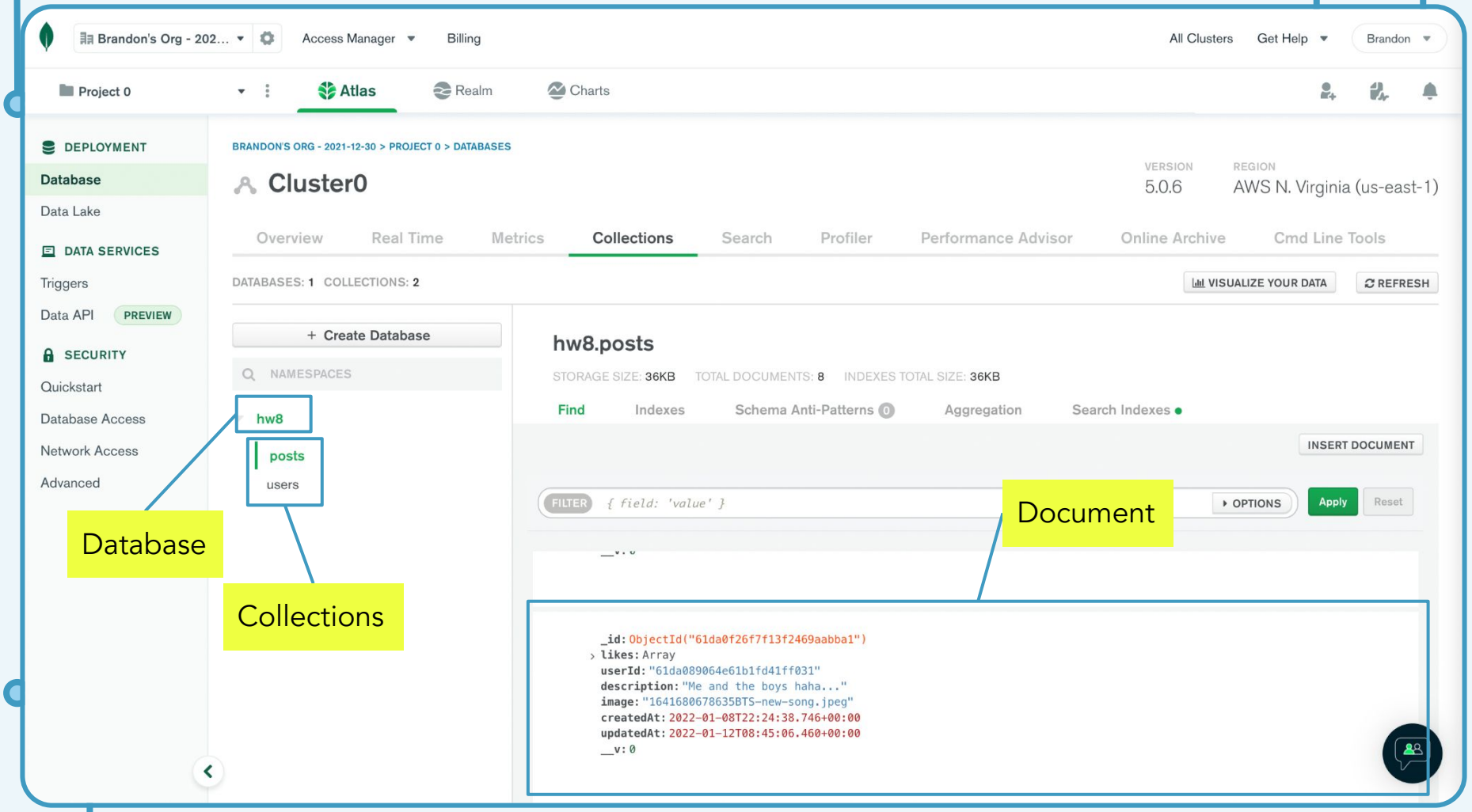


Analogy: Shelf -> Binders -> Pages

Example: Spotify



MongoDB Web Interface


Brandon's Org - 202...
Access Manager
Billing
All Clusters
Get Help
Brandon
Project 0
Atlas
Realm
Charts
DEPLOYMENT
Database
Data Lake
DATA SERVICES
Triggers
Data API
PREVIEW
SECURITY
Quickstart
Database Access
Network Access
Advanced
BRANDON'S ORG - 2021-12-30 > PROJECT 0 > DATABASES
Cluster0
VERSION
5.0.6
REGION
AWS N. Virginia (us-east-1)
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 1
COLLECTIONS: 2
VISUALIZE YOUR DATA
REFRESH
+ Create Database
NAMESPACES
hw8
posts
users
hw8.posts
STORAGE SIZE: 36KB
TOTAL DOCUMENTS: 8
INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER
{ field: 'value' }
OPTIONS
Apply
Reset
Database
Collections
Document
_id: ObjectId("61da0f26f7f13f2469aabba1")
likes: Array
userId: "61da089064e61b1fd41ff031"
description: "Me and the boys haha..."
image: "1641680678635BTS-new-song.jpeg"
createdAt: 2022-01-08T22:24:38.746+00:00
updatedAt: 2022-01-12T08:45:06.460+00:00
__v: 0

Mongoose

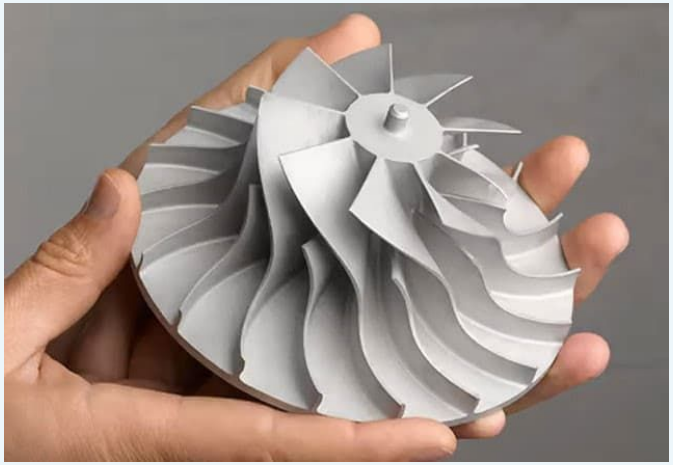
Mongoose overview

mongoose

- For our purposes, it is a JavaScript library for communicating with MongoDB (An abstraction on top of MongoDB to make it easier)
- You can query the database, make changes, and save changes via mongoose
- Mongoose can also model objects, like the structure of a particular document within a collection

Mongoose overview

mongoose



Blueprint → Prototype → Actual Thing!

Schema → Model → Document

Mongoose overview

mongoose

Schemas define the structures and properties of a MongoDB document

```
const kittySchema = new mongoose.Schema({ // Schema
  name: String
});
```

Blueprint

Schema

Each key in our code `kittySchema` defines a property in our document which will be cast to its associated SchemaType. For example, we've defined a property `name` which will be cast to the String SchemaType.

Mongoose overview

mongoose

Schemas define the structures and properties of a MongoDB document

```
const kittySchema = new mongoose.Schema({ // Schema
  name: String
});
```

Blueprint

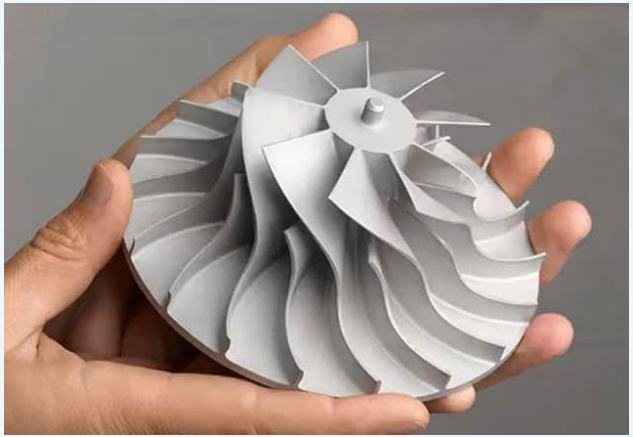
Schema

Here are a few permitted SchemaTypes:

- String
- Number
- Boolean
- Array

Mongoose overview

mongoose



Models are compiled versions of Schemas that handle database operations such as creating, querying, updating, and deleting data in a collection

```
const kittySchema = new mongoose.Schema({ // Schema
  name: String
});

const Kitten = mongoose.model('Kitten', kittySchema);
```

Collection name

Schema name

Prototype

Model

Mongoose overview

mongoose

Actual Thing!

Document

Documents store your actual data! They are instances of your model.

```
const kittySchema = new mongoose.Schema({ // Schema
  name: String
});

const Kitten = mongoose.model('Kitten', kittySchema);

const silence = new Kitten({ name: 'Silence' }); // Do
const fluffy = new Kitten({ name: 'fluffy' }); // DOcu

await silence.save()
await fluffy.save()
```

Put it all together!

```
async function main() {
  await mongoose.connect('mongodb://localhost:27017/test');
}
```

Connect your web app to MongoDB

```
const kittySchema = new mongoose.Schema({ // Schema
  name: String
});
```

Create a schema

```
const Kitten = mongoose.model('Kitten', kittySchema); // Mode
```

Create a model

```
const silence = new Kitten({ name: 'Silence' }); // Document
const fluffy = new Kitten({ name: 'fluffy' }); // DOcument
```

Create documents

```
await silence.save()
await fluffy.save()
```

Save documents to your database

Demo

Mongoose and Express: POST requests

```
app.post("/kittens/new", async (req, res) => {
  const kitten = new Kitten({
        name: req.body.name
    })
    await kitten.save()
    res.json(kitten)
});
```

Mongoose and Express: GET requests

```
app.get("/kittens", async (req, res) => {
  const allKittens = await Kitten.find();
  return res.json(allKittens);
});

app.get("/kittens/:id", async (req, res) => {
  const kitten = await Kitten.findById(req.params.id);
  return res.json(kitten);
});
```

[end recording]

Attendance: Lecture 7

https://forms.gle/LAAZ28LAEzEcpfP59

Secret word: