CRUD Operations using Mongoose and MongoDB

So, in this section, you learned that:

- MongoDB is an open-source document database. It stores data in flexible, JSONlike documents.
- In relational databases we have **tables** and **rows**, in MongoDB we have **collections** and **documents**. A document can contain sub-documents.
- We don't have relationships between documents.
- To connect to MongoDB:

```
// Connecting to MongoDB const mongoose =
require('mongoose');
mongoose.connect('mongodb://localhost/playground')
  .then(() => console.log('Connected...'))
  .catch(err => console.error('Connection failed...'));
```

- To store objects in MongoDB, we need to define a Mongoose **schema** first. The schema defines the shape of documents in MongoDB.

```
// Defining a schema const courseSchema = new
mongoose.Schema({       name: String,       price: Number

});
```

- We can use a SchemaType object to provide additional details:

```
// Using a SchemaType object const courseSchema = new
mongoose.Schema({        isPublished: { type: Boolean,
default: false }
});
```

- Supported types are: **String, Number, Date, Buffer** (for storing binary data), **Boolean** and **ObjectID**.
- Once we have a schema, we need to compile it into a model. A model is like a class. It's a blueprint for creating objects:

```
// Creating a model
const Course = mongoose.model('Course', courseSchema);
```

CRUD Operations

```
// Saving a document let course = new
Course({ name: '...' }); course = await
course.save();

// Querying documents const
courses = await Course
  .find({ author: 'Yagnesh, isPublished: true })
  .skip(10)
  .limit(10)
  .sort({ name: 1, price: -1 })
  .select({ name: 1, price: 1 });
```

```
// Updating a document (query first) const course
= await Course.findById(id); if (!course) return;
course.set({ name: '...' }); course.save();

// Updating a document (update first) const result =
await Course.update({ _id: id }, {
  $set: { name: '...' }
});

// Updating a document (update first) and return it const result = await
Course.findByIdAndUpdate({ _id: id }, {
  $set: { name: '...' }
}, { new: true });

// Removing a document const result = await
Course.deleteOne({ _id: id }); const result = await
Course.deleteMany({ _id: id }); const course = await
Course.findByIdAndRemove(id);
```