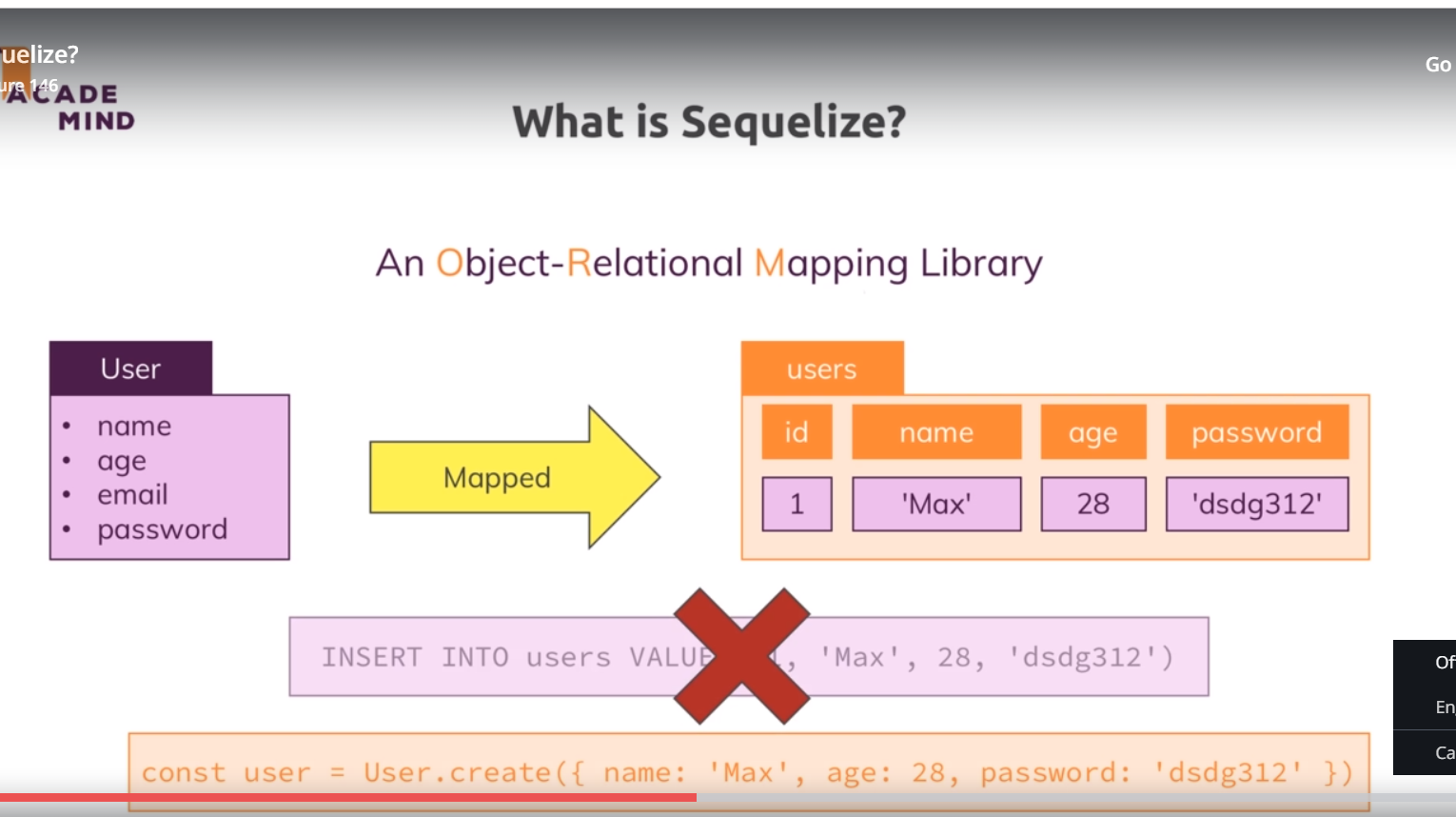
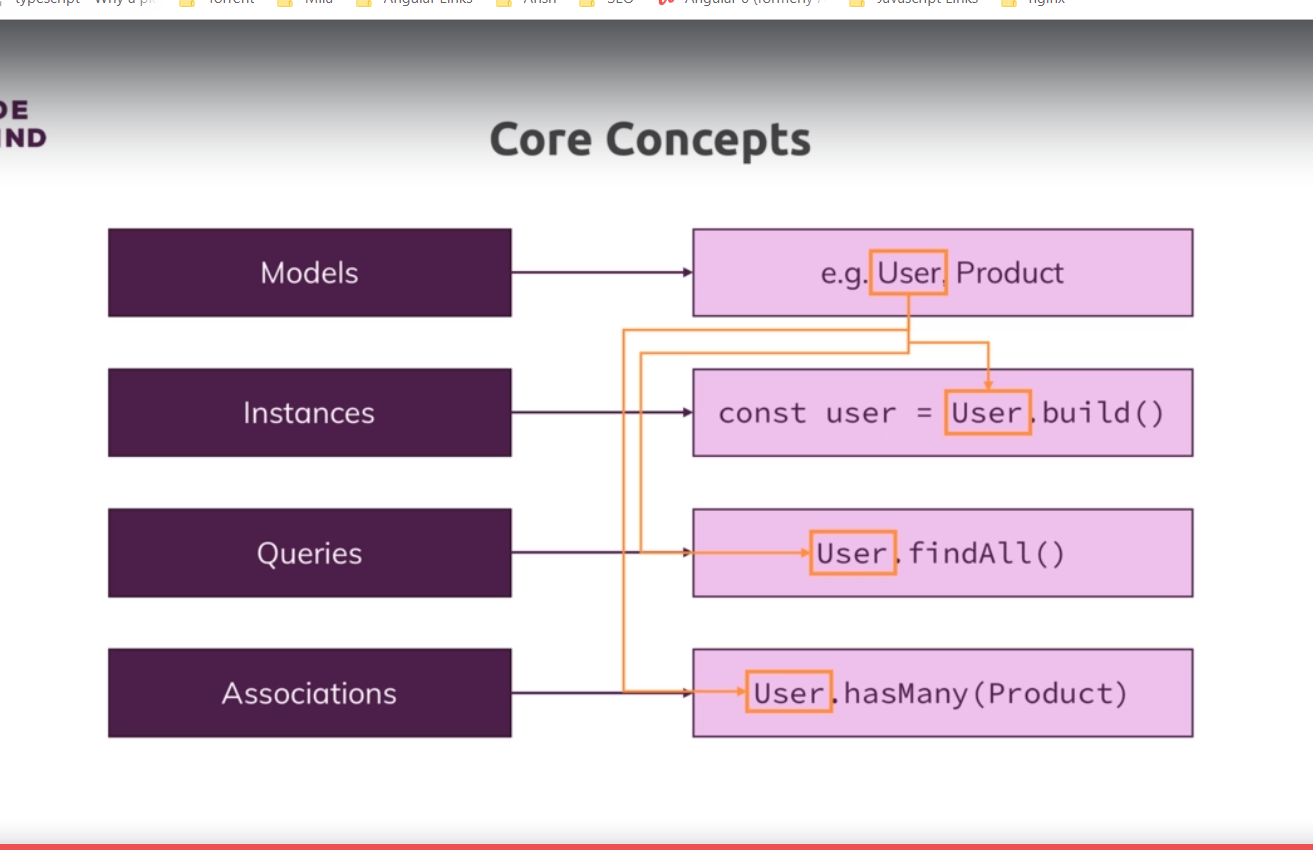
146)what is sequelize



We dnt need to write SQl queries on our own, we can just call a method on js object and seqalize will write sql queries in backend for us.

Seqalize offers us models to work with db as I showed you in last slide and it allows us to define such models, so basically define which data makes up model and therefore which data will be saved in database. Then we can instantiate these models, so these classes so to say, we can execute the constructor functions ot use utility methods to create lets say a new user object based on that model.then we can queries on that. We can also associate our user model to a product model. So this is what sequalize does.



147)Connecting to database  
lets install sequalize package. Run this command-

**npm install --save sequelize**

we also need mysql2 package that we installed in last lecture.

Now first step is that we need to create a model , and well alsow e need to connect with database.

Delete the products table that you have created in last lecture. right click on table -> drop

Then select drop now. We do this because now we want to use sequalize to manage our database. We connect to database in utils/database.js file. here we have connect to db isg sql2 packag, code-

const mysql = require('mysql2');

const pool = mysql.createPool({

host: 'localhost',

user: 'root',

database: 'node-complete',

password: 'sumeet'

});

module.exports = pool.promise();

now sequalize uses mySQL2 behind the scenes, therefore sequalize behind the scenes will use something like above code, but we wnt write that. This is how we connect with database using sequalize-

const Sequelize = require('sequelize');

const seqelize = new Sequelize('node-complete','root','sumeet',{dialect: 'mysql', host: 'localhost'});

module.exports = seqelize;

here fourth argument is optional. use can use type {}, take cursor between them and press  **ctrl+ space** to see various options. Here we use dialect option, it tells that we want to specifically want to connect with mysql. Different db’s have slightly different sql syntax. Then we set host, by default it is **localhost** we dnt to set it. But for being explicit here we set it.

With that we are creating new sequelize object and it will automatically connect to the database then or to be precise it will setup the connection pool just as we did it manually in last module. Then we export sequalize object which is essentially that connection pool ,managed by sequalize giving us a lot of useful features. With that connection is established now lets next work on model.

148)Defining a Model

In models/product.js, remove all code. We import 2 things, first we import Sequelize, that will give me back class or constructor function, so we use capital S. the next thing I will import is my database connection pool managed by sequelize, we also name it sequalize but with lower case s. with these 2 things , w ecan now define model which will be mnaged by sequelize. Then we define, product, this is not defined as class now, as we didi before. Instead now we use sequalize, our database connection pool which is more than a connection pool, it’s actually a fully configured sequelize envirenment which does also have connection pool but also has all the feature of seqelize. Now we define new model by calling define method on it.

You can hover over define to see which options you have on it. First argument is model name. now model name typically is lower case name, so we use  **product** as model name. the second argument defines the structure of our model and also structure of automatically created database table. It is js object, here we define attributes or fields our product should have. Like our product should have id. Then we gain define id with a js object. here we define type of id. It needs to be one of types define by sequalize package. This is here we need our first Import. typeSequalize. , ide will tell you all avalaible types. We can see number, strings(nor varchar, here we have more javacsriptish types)

For details go to docs of sequqlize. Oh left hand side click on model definition, here you will find all options.that you can use while defining column. Link-

<http://docs.sequelizejs.com/manual/tutorial/models-definition.html>

this is how we define our model-

models/product.js-

const Sequelize = require('sequelize');

const sequelize = require('../util/database');

const Product = sequelize.define('product',{

id: {

type: Sequelize.INTEGER,

autoIncrement: true,

allowNull: false,

primaryKey: true

},

title: Sequelize.STRING,

price: {

type: Sequelize.DOUBLE,

allowNull: false

},

imageUrl: {

type: Sequelize.STRING,

allowNull: false

},

description: {

type: Sequelize.STRING,

allowNull: false

}

});

module.exports = Product;

each column is described by object. if we just want to define datatype then we can also use shortcut. Like we did for title.

At last we export Product. Now lets start using it.

149)Syncing Js Definations to Database

We created product model and I mentioned thatw ecan now start working with tis model to create products and so on. But for this we ofcourse lso need product table in our database. We deleted all tables. Now sequalize can create tables for us.

We do that app.js file. I want to ensure that all models are basically transferred into tables or get a table that belongs to them whenever we start the application and if table exist, it will not ofcourse override it by default althrough we can tell it to do so.

In app.js we import seqalize and on it we call special method called sync. The sync method has a look at all the models you defined and then basically creates tables for them. It is what sync does, it syncs your models to the database by creating the appropriate tables and relations. Sync returns the promise and we want to start our server only if we somehow makes into then. code-

App.js-

const sequelize = require('./util/database');

sequelize.sync()

.then(result => {

console.log(result);

app.listen(3000);

})

.catch(err => console.log(err));

We printed the result, it prints some query to create table, with all the fleds thatw e have mentioned in our model. Now if we open workbench, refresh the database. We can see our table there.