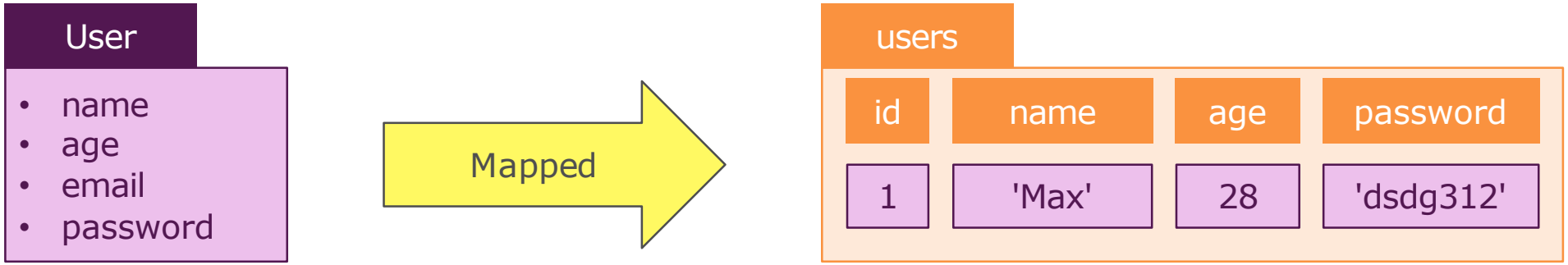
Mongoose

What is Mongoose?

- Mongoose ODM is an object document modeling package for Node that essentially works like an ORM (like Hibernate in Java).
- Mongoose allows us to have access to the MongoDB commands for CRUD simply and easily.




```
db.collection('users').insertO    ame: 'Max', age: 28, password:
                          'ds      ' })
```

```
const user = User.create({ name: 'Max', age: 28, password: 'dsdg312' })
```

Getting Started with Mongoose

```
// install mongoose package
$ npm install mongoose --save

// import the library in your application
const mongoose = require('mongoose');

// connect to a MongoDB database
mongoose.connect('mongodb://localhost:27017', { useNewUrlPa
rser: true, useUnifiedTopology: true })
    .then(() => {
        …
    }).catch(err => console.error(err));
```

Defining a Model

- Mongoose Schema is what we use to define attributes for our documents (structure of the document).

- Before we can handle CRUD operations, we will need a mongoose Model. These models are constructors that we define. They represent documents which can be saved and retrieved from our database.

- Mongoose Methods can also be defined on a mongoose schema.

Working with Mongoose

1. **Create schema**

```
var mySchema = new mongoose.Schema({ name: String });
```

2. **Convert schema to model** (collection)

```
var Collection = mongoose.model('Collection', mySchema);
```

3. **Every instance of the model represents a document**

```
var doc = new Collection({ name: 'Josh' });
```

Mongoose - Model Example & Create

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const productSchema = new Schema({
    title: {
        type: String,
        required: true
    },
    price: {
        type: Number,
        required: true,
    },
    description: String,
    imageUrl: String
});

//model name will be used to turn into collection name
//'Product' -> lower case 'product' + 's'
module.exports = mongoose.model('Product', productSchema);
```


https://mongoosejs.com/docs/guide.html

Mongoose – CRUD

```
const Product = require('../models/product');

exports.getProducts = (req, res, next) => {
    Product.find()
        .then(products => {
            res.render('admin/products', {
                pageTitle: 'Products',
                path: '/admin/products',
                prods: products
            });
        })
        .catch(err => console.log(err));
};

exports.getAddProduct = (req, res, next) => {
    res.render('admin/add-product', {
        pageTitle: 'Add Product',
        path: '/admin/add-product'
    });
};
```

```
exports.postAddProduct = (req, res, next) => {
    const title = req.body.title;
    const imageUrl = req.body.imageUrl;
    const price = req.body.price;
    const description = req.body.description;
    const product = new Product({
        title: title,
        price: price,
        description: description,
        imageUrl: imageUrl
    });
    product.save()
        .then(result => {
            res.redirect('/');
        })
        .catch(err => console.log(err));
};
```

Mongoose – CRUD (cont.)

```
exports.getEditProduct = (req, res, next) => {
    const prodId = req.params.prodId;
    Product.findById(prodId)
        .then(product => {
            res.render('admin/edit-product', {
                product: product,
                pageTitle: 'Edit Product',
                path: '/admin/products',
            });
        })
        .catch(err => console.log(err));

}
exports.postDeleteProduct = (req, res, next) => {
    Product.findByIdAndRemove(req.body._id)
        .then(result => {
            res.redirect('/admin/products');
        })
        .catch(err => console.log(err));
}
```

```
exports.postEditProduct = (req, res, next) => {
    const id = req.body._id;
    const title = req.body.title;
    const imageUrl = req.body.imageUrl;
    const price = req.body.price;
    const description = req.body.description;

    Product.findById(id).then(prod => {
        prod.title = title;
        prod.imageUrl = imageUrl;
        prod.price = price;
        prod.description = description;
        return prod.save();
    })
        .then(result => {
            res.redirect('/admin/products');
        })
        .catch(err => console.log(err));

}
```

Working with Relations

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const userSchema = new Schema({
    name: {
        type: String,
        required: true,
    },
    email: {
        type: String,
        required: true,
    },
    cart: {
        items: [{
            productId: { type: Schema.Types.ObjectId, ref: 'Product', required: true },
            quantity: { type: Number, required: true }
        }]
    }
});
module.exports = mongoose.model('User', userSchema);
```

Instance Methods

- Instances of Models are documents. Documents have many of their own built-in instance methods. We may also define our own custom document instance methods too.

```
userSchema.methods.deleteItemFromCart = function (productId) {
    const updatedCartItems = this.cart.items.filter(item => {
        return item.productId.toString() !== productId.toString();
    });
    this.cart.items = updatedCartItems;
    return this.save();
}
```

Populating an existing document

- If we have one or many mongoose documents or even plain objects (like mapReduce output), we may populate them using the Model.populate() method available in mongoose >= 3.6. This is what document#populate() and query#populate() use to populate documents.

```
exports.getCart = (req, res, next) => {
    req.user
        .populate('cart.items.productId')
        .execPopulate()
        .then(user => {
            const products = user.cart.items;
            res.render('shop/cart', {
                path: '/cart',
                pageTitle: 'Your Cart',
                products: products
            });
        })
        .catch(err => console.log(err));
};
```

Statics

- You can also add static functions to your model. There are 2 equivalent ways to add a static:
  - Add a function property to schema.statics
  - Call the Schema#static() function

```
// Assign a function to the "statics" object of our animalSchema
animalSchema.statics.findByName = function (name) {
    return this.find({ name: new RegExp(name, 'i') });
};
// Or, equivalently, you can call `animalSchema.static()`.
animalSchema.static('findByBreed', function (breed) {
    return this.find({ breed });
});
```

Virtuals

- Virtuals are document properties that you can get and set but that do not get persisted to MongoDB. The getters are useful for formatting or combining fields, while setters are useful for de-composing a single value into multiple values for storage.

```
UserSchema
    .virtual('name')
    .get(function () {
        return this.last_name + ', ' + this.first_name;
    });
```

Resources

- Mongoose Resources
  - Mongoose
  - Mongoose Documentation
  - Mongoose Schemas
  - Mongoose Models
  - Mongoose Sub-documents
  - Mongoose-currency

Homework

- Continue working on our project, add features below:
  - CRUD on Shopping Cart
  - Place an Order and display orders.