MONASH University

FIT2095 e-Business software technologies - S2 2021

Question **1**

Not yet answered

Marked out of 1.00

```
let author1 = new Author({
       _id: new mongoose.Types.ObjectId(),
       name: {
            firstName: 'Tim',
            lastName: 'John'
       },
       age: 80
 });

 author1.save(function (err) {
        if (err) throw err;

        console.log('Author successfully Added to DB');
  };

var book1 = new Book({
      _id: new mongoose.Types.ObjectId(),
      title: 'FIT2095 Book ',
      author: author1._id,
       isbn: '123456',
   });

book1.save(function (err) {
 if (err) throw err;

 console.log('Book1 successfully Added to DB');
 });
```

With respect to the presented code, which of the following is true?

Select one:

[✓ a.] The code has an issue.

The creating and saving of book1 should be done inside author1.save() callback function due to the asynchronous call of I/O operations

b. The code is perfect and has no issue

c. The code has an issue.

Both author1 and book1 will be saved in the same collection.

d. The code a syntax error.

It uses 'var' instead of 'let'

Clear my choice

Question 2

Not yet answered

Marked out of 1.00

Develop a schema for two properties: itemName (String) and quantity (Number) where the item quantity is a positive number.

Which of the following statements fulfills the above requirements and is correct?

Select one:

[✓ a.]

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: {
          type: Number,
          validate: {
                    validator: function (newQuantity) { return newQuantity >= 0;},
                    message: 'The value is not positive'
                  }
           }
  });
```

b.

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: {
          type: Number,
          validate: {
                    validator: function (newQuantity) { if(newQuantity >= 0) console.log('It is not positive);}
                  }
           }
  });
```

c.

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: {
          type: Number,
          validate: {
                    function (newQuantity) { return newQuantity >= 0;},
                    message: 'The value is not positive'
                  }
           }
  });
```

d.

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: {
          type: Number,
          validate: function (newQuantity) { return newQuantity >= 0};
      }
});
```

e.

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: {
          type: Number,
          validate: {
                    validator: function (newQuantity) { return newQuantity < 0;},
                    message: 'The value is not positive'
                  }
           }
  });
```

Clear my choice

Question **3**

Not yet answered

Marked out of 1.00

Assume you have this Mongoose URL:

```
let url='mongodb://localhost:27017/Travel';
```

What does the word 'Travel' represent?

Select one:

- ○ a. last part of the server's address
- ○ b. MongoDB replica name
- ○ c. Collection name
- [✓ ● d. Database name]

Clear my choice

Question **4**

Not yet answered

Marked out of 1.00

```
let author1 = new Author({
 _id: new mongoose.Types.ObjectId(),
 name: {
      firstName: 'Tim',
      lastName: 'John'
   },
 age: 80
});
```

Find all the documents with the first name equals 'Tim'. Which of the following is correct?

Select one:

○ a.
```
Items.find({ 'name.firstName': /^Tim/ }, function (err, docs) {
   //Do something
});
```

○ b.
```
Items.find({ name.firstName: 'Tim' }, function (err, docs) {
   //Do something
});
```

○ c.
```
Items.find({ 'name.firstname': 'Tim' }, function (err, docs) {
   //Do something
});
```

[✓ ● d.]
```
Items.find({ 'name.firstName': 'Tim' }, function (err, docs) {
   //Do something
});
```

○ e.
```
Items.findOne({ 'firstName': 'Tim' }, function (err, docs) {
   //Do something
});
```

Clear my choice

Question 5

Not yet answered

Marked out of 1.00

Retrieve the first 50 documents with quantity between 100 and 150 inclusive.

Which of the following statements fulfills the above and is correct?

Select one:

○ a.
```
Items.where('quantity').gte(100).lte(150).limit(50).function (err, docs) {
 // Do something with Docs
};
```

[✓ b.]
```
Items.where('quantity').gte(100).lte(150).limit(50).exec(function (err, docs) {
 // Do something with Docs
});
```

○ c.
```
Items.where('quantity').gt(100).lt(150).limit(50).exec(function (err, docs) {
 // Do something with Docs
});
```

○ d.
```
Items.where('quantity').gte(100).lte(150).sort(50).exec(function (err, docs) {
 // Do something with Docs
});
```

Clear my choice

Question 6

Not yet answered

Marked out of 1.00

The following statement creates a new Model for the 'itemSchema' schema.

```
module.exports = mongoose.model('Items', itemSchema);
```

What does 'Items' represent?

Select one:

○ a. The name of the server that hosts the database

○ b. The name of the primary key column

○ c. The name of the items' collection

[✓ d. Just a unique name to identify the schema]

○ e. The name of the database the schema will connect to

Clear my choice

Question 7

Not yet answered

Marked out of 1.00

Develop a Mongoose schema that consists of two properties: item name (string) and quantity (integer). The quantity should be required and has a default value = 0.

which of the following fulfills the above requirements and correct?

Select one:

○ a.
```
var itemSchema = mongoose.Schema({
   _id: mongoose.Schema.Types.Id,
   itemName: String,
   quantity: {
          type: Integer,
          required: true,
          default: 0
        }
  });
```

○ b.
```
var itemSchema = mongoose.Schema({
   _id: mongoose.Schema.Types.ObjectId,
   itemName: String,
   quantity: {
          type: Integer,
          required: true,
          default: 0
        }
  });
```

○ c.
```
var itemSchema = mongoose.Schema({
   _id: mongoose.Schema.Types.ObjectId,
   itemName: String,
   quantity: {
          type: Number,
          required: 'True',
          default: 0
        }
  });
```

[✓ d.]
```
var itemSchema = mongoose.Schema({
   _id: mongoose.Schema.Types.ObjectId,
   itemName: String,
   quantity: {
          type: Number,
          required: true,
          default: 0
        }
  });
```

○ e.
```
var itemSchema = MongoDB.Schema({
   _id: mongoose.Schema.Types.ObjectId,
   itemName: String,
   quantity: {
          type: Number,
          required: true,
          default: 0
        }
  });
```

Clear my choice

Question **8**

Not yet answered

Marked out of 1.00

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js.

With respect to Mongoose, which of the following statements is false?

Select one:

○ a. With Mongoose, I am able to implement One-To-Many relationship between collections

○ b. Query functions -such as findOne and UpdateMany- are accessible through Models, not schemas.

○ c. Using Mongoose, you can strongly-typed MongoDB collections.

[✓ d. In order to use Mongoose, you must have a reference to MongoDB module.]

```
const mongodb=require('MongoDB');
const mongoose = require('mongoose');
```

○ e. Using Mongoose, you can build your own validation function.

Clear my choice

Question **9**

Not yet answered

Marked out of 1.00

Which of the following is false about the presented code?

```
let bookSchema = mongoose.Schema({
 _id: mongoose.Schema.Types.ObjectId,
     title: {
             type: String,
             required: true
           },
     isbn: String,
     author: {
             type: mongoose.Schema.Types.ObjectId,
             ref: 'Author'
           },
     created: {
            type: Date,
            default: Date.now
          }
});
```

Select one:

○ a. the field isbn is optional

○ b. the presented code declares a schema without a Model

[✓ c. the field **created** is mandatory]

○ d. the field author is a reference to another schema (document)

Clear my choice

Question **10**

Not yet answered

Marked out of 1.00

Develop a schema for items with two properties: itemName (String) and quantity (Integer). The quantity should be saved if it is Integer (not decimal).

Which of the following fulfills the above requirements and is correct?

Hint: https://www.w3schools.com/jsref/jsref_isinteger.asp

Select one:

○ a. Declaring the quantity field data type as Number would be enough.

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: Number
});
```

[✓ b.]

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: {
        type: Number,
        validate: {
              validator: function(quantity){Number.isInteger(quantity);},
              message: 'The quantity is not an integer value'
             }
     }
});
```

○ c.

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: {
        type: Number,
        validate: {
              validator: Number.isInteger,
              message: 'The quantity is not an integer value'
             }
     }
});
```

○ d.

```
var itemSchema = mongoose.Schema({
    _id: mongoose.Schema.Types.ObjectId,
    itemName: String,
    quantity: {
        type: Number,
        validate: {
              validator: isInteger,
              message: 'The quantity is not an integer value'
             }
     }
});
```

Clear my choice

◄ Week 5 Pre-Reading Quiz

Jump to...

Lab Week 2 ►