Create Operations

Create or insert operations add new [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) to a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections). If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

* [db.collection.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne) *New in version 3.2*
* [db.collection.insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany) *New in version 3.2*

In MongoDB, insert operations target a single [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection). All write operations in MongoDB are [atomic](https://docs.mongodb.com/manual/core/write-operations-atomicity/) on the level of a single [document](https://docs.mongodb.com/manual/core/document/).

For examples, see [Insert Documents](https://docs.mongodb.com/manual/tutorial/insert-documents/).

Read Operations

Read operations retrieve [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) from a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections); i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:

* [db.collection.find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)

You can specify [query filters or criteria](https://docs.mongodb.com/manual/tutorial/query-documents/#read-operations-query-argument) that identify the documents to return.

click to enlarge

For examples, see:

* [Query Documents](https://docs.mongodb.com/manual/tutorial/query-documents/)
* [Query on Embedded/Nested Documents](https://docs.mongodb.com/manual/tutorial/query-embedded-documents/)
* [Query an Array](https://docs.mongodb.com/manual/tutorial/query-arrays/)
* [Query an Array of Embedded Documents](https://docs.mongodb.com/manual/tutorial/query-array-of-documents/)

Update Operations

Update operations modify existing [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) in a [collection](https://docs.mongodb.com/manual/core/databases-and-collections/#collections). MongoDB provides the following methods to update documents of a collection:

* [db.collection.updateOne()](https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#db.collection.updateOne) *New in version 3.2*
* [db.collection.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany) *New in version 3.2*
* [db.collection.replaceOne()](https://docs.mongodb.com/manual/reference/method/db.collection.replaceOne/#db.collection.replaceOne) *New in version 3.2*

In MongoDB, update operations target a single collection. All write operations in MongoDB are [atomic](https://docs.mongodb.com/manual/core/write-operations-atomicity/) on the level of a single document.

You can specify criteria, or filters, that identify the documents to update. These [filters](https://docs.mongodb.com/manual/core/document/#document-query-filter) use the same syntax as read operations.

For examples, see [Update Documents](https://docs.mongodb.com/manual/tutorial/update-documents/).

Delete Operations

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

* [db.collection.deleteOne()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteOne/#db.collection.deleteOne) *New in version 3.2*
* [db.collection.deleteMany()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/#db.collection.deleteMany) *New in version 3.2*

In MongoDB, delete operations target a single [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection). All write operations in MongoDB are [atomic](https://docs.mongodb.com/manual/core/write-operations-atomicity/) on the level of a single document.

You can specify criteria, or filters, that identify the documents to remove. These [filters](https://docs.mongodb.com/manual/core/document/#document-query-filter) use the same syntax as read operations.

For examples, see [Delete Documents](https://docs.mongodb.com/manual/tutorial/remove-documents/).

## Insert a Single Document

[db.collection.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne) inserts a single [document](https://docs.mongodb.com/manual/core/document/#bson-document-format) into a collection.

The following example inserts a new document into the inventory collection. If the document does not specify an \_id field, MongoDB adds the \_id field with an ObjectId value to the new document. See [Insert Behavior](https://docs.mongodb.com/manual/tutorial/insert-documents/#write-op-insert-behavior).

db.inventory.insertOne(

{ item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }

)

You can run the operation in the web shell below:

[insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne) returns a document that includes the newly inserted document’s \_id field value. For an example of a return document, see [db.collection.insertOne() reference](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/" \l "insertone-examples).

To retrieve the document that you just inserted, [query the collection](https://docs.mongodb.com/manual/core/document/#document-query-filter):

db.inventory.find( { item: "canvas" } )

## Insert Multiple Documents

*New in version 3.2.*

[db.collection.insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany) can insert multiple [documents](https://docs.mongodb.com/manual/core/document/#bson-document-format) into a collection. Pass an array of documents to the method.

The following example inserts three new documents into the inventory collection. If the documents do not specify an \_id field, MongoDB adds the \_id field with an ObjectId value to each document. See [Insert Behavior](https://docs.mongodb.com/manual/tutorial/insert-documents/#write-op-insert-behavior).

db.inventory.insertMany([

{ item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },

{ item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },

{ item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }

])

You can run the operation in the web shell below:

[insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany) returns a document that includes the newly inserted documents \_id field values. See the [reference](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#insertmany-examples) for an example.

To retrieve the inserted documents, [query the collection](https://docs.mongodb.com/manual/tutorial/query-documents/#read-operations-query-document):

db.inventory.find( {} )

This is the code to use MongoDB native driver:

const MongoClient = require('mongodb').MongoClient;

const assert = require('assert');

*// Connection URL*

const url = 'mongodb://localhost:27017';

*// Database Name*

const dbName = 'fruitsDB';

const client = new MongoClient(url);

*// Use connect method to connect to the server*

client.connect(function(err) {

  assert.equal(null, err);

  console.log('Connected successfully to server');

  const db = client.db(dbName);

*// client.close();*

  findDocuments(db, function() {

*// client.close();*

  });

});

const insertDocuments = function(db, callback) {

*// Get the documents collection*

  const collection = db.collection('fruits');

*// Insert some documents*

  collection.insertMany([

    {

      name:"Apple",

      score: "6",

      review:"Great fruit",

    },

    {

      name:"Orange",

      score: "7",

      review:"Sour",

    },

    {

      name:"Banana",

      score: "9",

      review:"Noice!",

    }

  ], function(err, result) {

    assert.equal(err, null);

    assert.equal(3, result.result.n);

    assert.equal(3, result.ops.length);

    console.log('Inserted 3 documents into the collection');

    callback(result);

  });

};

const findDocuments = function(db, callback) {

*// Get the documents collection*

  const collection = db.collection('fruits');

*// Find some documents*

  collection.find({}).toArray(function(err, fruits) {

    assert.equal(err, null);

    console.log("Found the following records");

    console.log(fruits)

    callback(fruits);

  });

}

Mongoose:

We can mongoose to simplify and reduce the boilerplate code. It is an object document mapper.

Notes are mentioned in comments in project.