MongoDB

Specific search:

You can also add search parameters to find specific documents.

Example: find documents whose name is Juhi.

Just change the query to this:

```
dbo.collection("students").find({name:'Juhi'}).toArray(function(err, result) {
or
var query = {name: "Juhi"};
  dbo.collection("students").find(query).toArray(function(err, result) {
```

Find specific column values:

Example-1: Display name field only whose standard is 1.

```
var MongoClient=require('mongodb').MongoClient;
var url="mongodb://localhost/";
MongoClient.connect(url,{useUnifiedTopology:true},function(err,db)
{
  if(err) throw err;
  var dbo=db.db("class_demo");
  var query={standard:1};
  dbo.collection("students").find(query).toArray(function(err,res){
    if(err) throw err;
    console.log(res);
  });
});
```

Example-2: Find name fields of all documents.

```
var query={standard:1};
  dbo.collection("students").find(query).toArray(function(err,res){
//dbo.collection("students").find({}).toArray(function(err,res){
if(err) throw err;
    var l=res.length;
    for(var i=0;i<l;i++)
    {
    console.log(res[i].name);
    }
});
```

Filter With Regular Expressions:

You can write regular expressions to find exactly what you are searching for.

Regular expressions can only be used to query strings.

To find only the documents where the "name" field starts with the letter "H", use the regular expression /^H/:

Example – 3 : Find documents where the name starts with the letter "H"

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("class_demo");
  var query = {name: /^H/ };
  dbo.collection("students").find(query).toArray(function(err, result) {
     if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Example-4: Find documents where the name ends with the letter "i"

```
var query = {name: /i$/ };
```

Example-5: Find documents where the name starts with J and ends with the letter "a"

```
var query = {name: /^J.*a$/ };
```

Sort the Result:

Use the sort() method to sort the result in ascending or descending order.
The sort() method takes one parameter, an object defining the sorting order.

Example-6: Sort the result alphabetically by name

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("c");
  var mysort = { name: 1 };
  dbo.collection("students").find().sort(mysort).toArray(function(err, result) {
     if (err) throw err;
    console.log(result);
  db.close();
  });
```

});

Sort Descending
Use the value -1 in the sort object to sort descending.

{ name: 1 } // ascending
{ name: -1 } // descending

Limit the Result:

To limit the result in MongoDB, we use the limit() method.

The limit() method takes one parameter, a number defining how many documents to return.

Example-7: Limit the result to only return 5 documents

```
dbo.collection("students").find().limit(5).toArray(function(err, result){
    if (err) throw err;
  console.log(result);
 db.close();
```

Drop Collection:

drop() method is used to delete collection in database.

The drop() method takes a callback function containing the error object and the result parameter which returns true if the collection was dropped successfully, otherwise it returns false.

Example

Delete the "students" table:

```
var MongoClient = require('mongodb').MongoClient;

var url = "mongodb://localhost/";

MongoClient.connect(url, function(err, db) {
```

```
  if (err) throw err;
  var dbo = db.db("c");
  dbo.collection("students").drop(function(err, result) {
    if (err) throw err;
    if (result) console.log("Collection deleted");
    db.close();
  });
});
```