EMERGING TECHNOLOGIES
PRACTICAL JOURNAL
TYBSC.IT

Name:
Class:
PRN NO:
Subject:
Course Code:

Certificate

Name:	Class:
Roll No:	Exam No:

Institution ____

This is certified to be the bonafide work of the student in the ____________________ Laboratory during the academic year 20 / 20 .

No. of practicals certified __________ out of __________ in the subject of ______________

....
Teacher In-charge

....
Examiner's Signature

....
Principal

Date: ....

Institution Rubber Stamp

(N.B: The candidate is expected to retain his/her journal till he/she passes in the subject.)

INDEX

PRACTICAL NO – 1

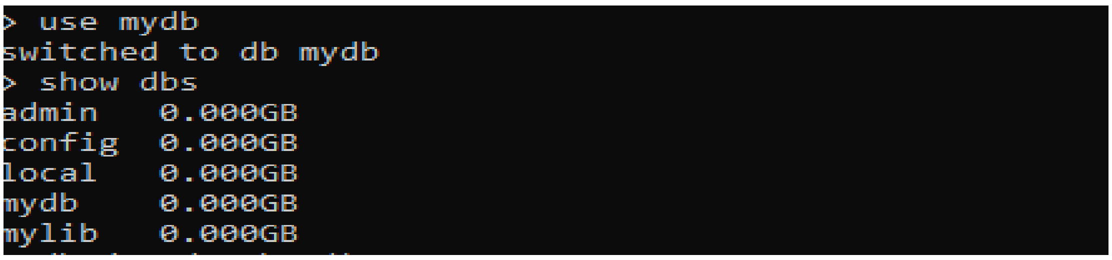
MONGODB BASICS

a. Write a MongoDB query to create and drop database.
b. Write a MongoDB query to create, display and drop collection.
c. Write a MongoDB query to insert, query, update and delete a document.

AIM 1.a) Write a MongoDB query to create and drop database.

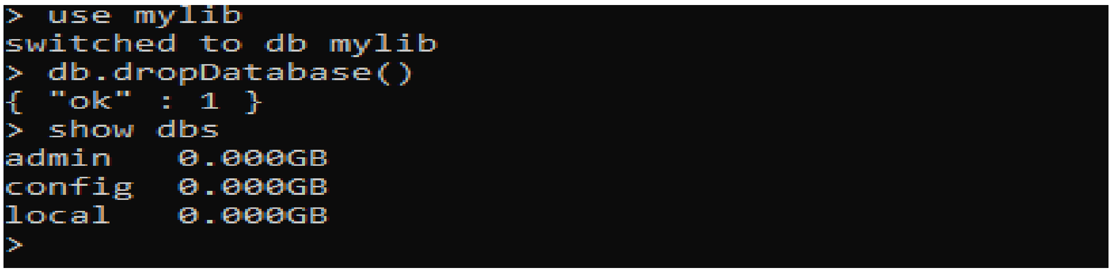
Syntax :
Create –
use database_name **Drop** – db.dropDatabase()
Source Code:
Use mydb db.dropDatabase()

OUTPUT :
Create database -



```
> use mydb
switched to db mydb
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
mydb     0.000GB
mylib    0.000GB
```

Drop Database –



```
> use mylib
switched to db mylib
> db.dropDatabase()
{ "ok" : 1 }
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
>
```

AIM 1.B) Write a MongoDB query to create, display and drop collection.

<u>Syntax</u>:

Create -

db.createCollection(collection_name) db.collection_name.insert({key:value})

Display – db.collection_name.find() **Drop -** db.collection_name.drop()

Source code:

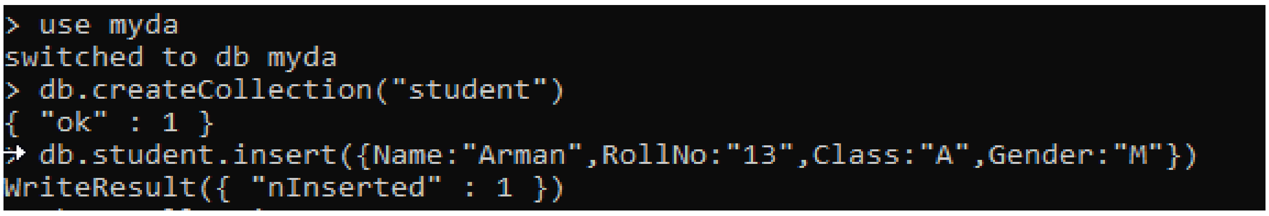
use myda

db.createCollection(“students”)

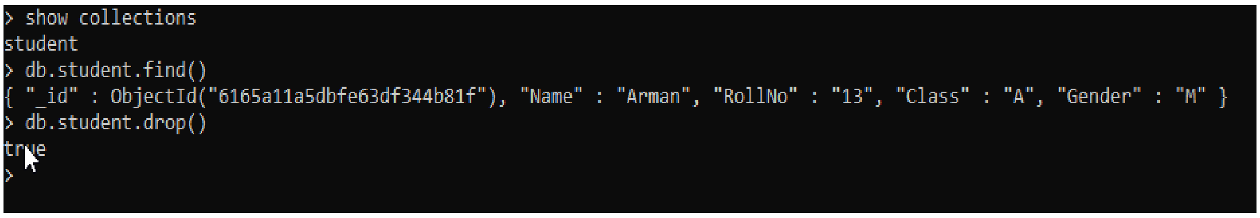
db.students.insert({Name:”Arman”,RollNo:”13”,Class:”A”,Gend er:”M”})

show collections db.students.find() db.students.drop()

OUTPUT :



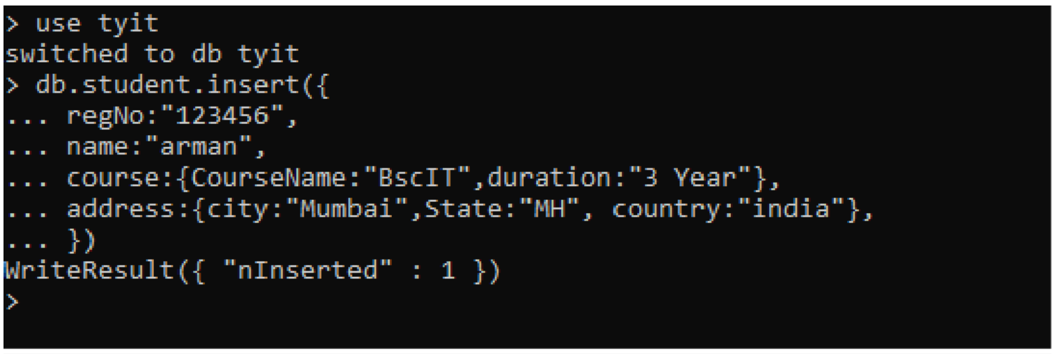
```
> use myda
switched to db myda
> db.createCollection("student")
{ "ok" : 1 }
> db.student.insert({Name:"Arman",RollNo:"13",Class:"A",Gender:"M"})
WriteResult({ "nInserted" : 1 })
```



```
> show collections
student
> db.student.find()
{ "_id" : ObjectId("6165a11a5dbfe63df344b81f"), "Name" : "Arman", "RollNo" : "13", "Class" : "A", "Gender" : "M" }
> db.student.drop()
true
>
```

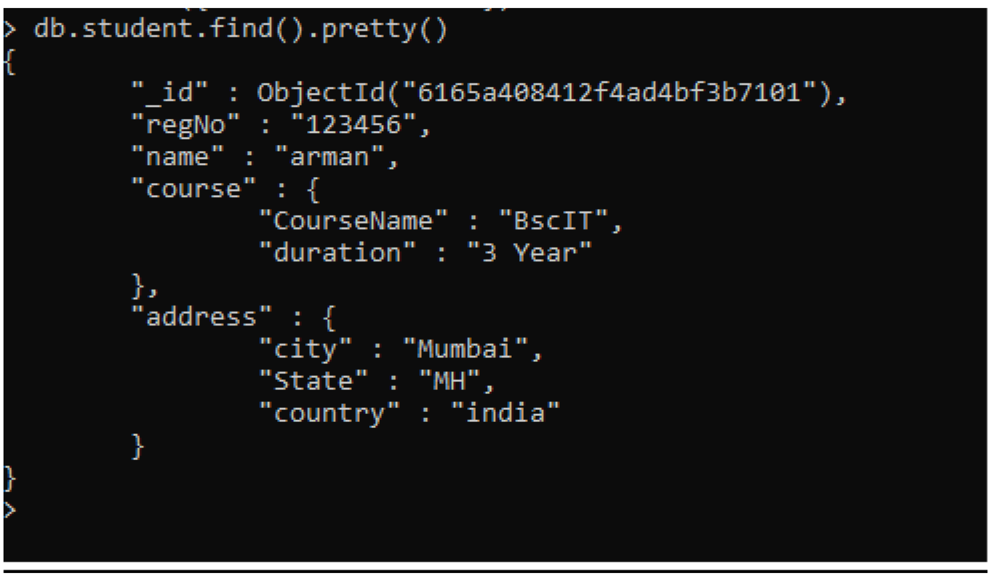
AIM 1.C) Write a MondoDB query to insert, query, update and delete a document.

Syntax:
To Insert Document: db.COLLECTION_NAME.insert(document) **To Query Document:** db.COLLECTION_NAME.ine()
To Update Document:
db.COLLECTION_NAME.update(SELECTION_CRITERIA,UPDATED DATA)
To Delete Document:
db.COLLECTION_NAME.remove(DELETION_CRITERIA)
OUTPUT :
Insert documentation -



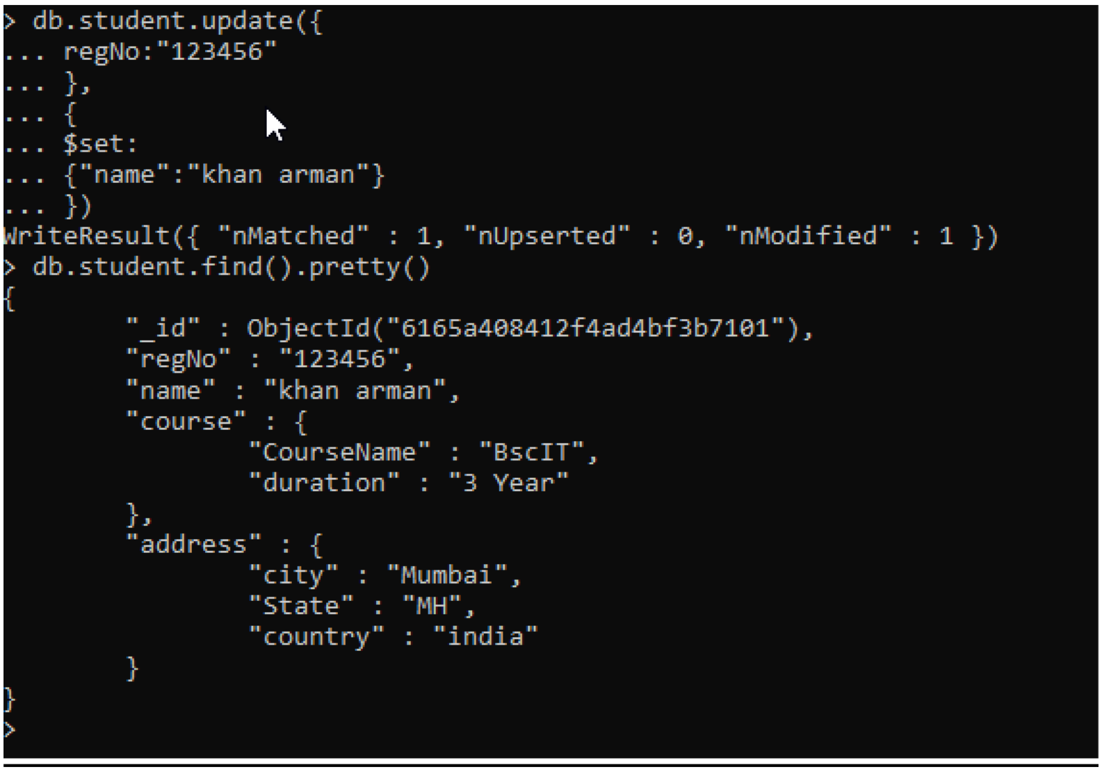
```
> use tyit
switched to db tyit
> db.student.insert({
... regNo:"123456",
... name:"arman",
... course:{CourseName:"BscIT",duration:"3 Year"},
... address:{city:"Mumbai",State:"MH", country:"india"},
... })
WriteResult({ "nInserted" : 1 })
>
```

Query Document –



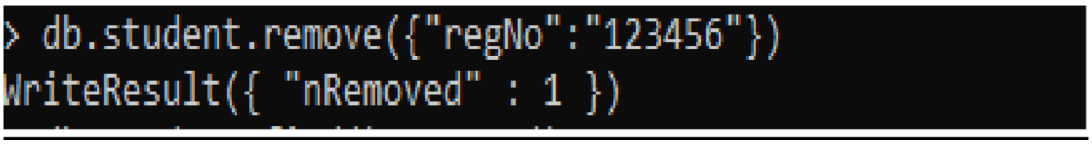
```
> db.student.find().pretty()
{
        "_id" : ObjectId("6165a408412f4ad4bf3b7101"),
        "regNo" : "123456",
        "name" : "arman",
        "course" : {
                "CourseName" : "BscIT",
                "duration" : "3 Year"
        },
        "address" : {
                "city" : "Mumbai",
                "State" : "MH",
                "country" : "india"
        }
}
>
```

UPDATE DOCUMENT -



```
> db.student.update({
... regNo:"123456"
... },
... {
... $set:
... {"name":"khan arman"}
... })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.student.find().pretty()
{
        "_id" : ObjectId("6165a408412f4ad4bf3b7101"),
        "regNo" : "123456",
        "name" : "khan arman",
        "course" : {
                "CourseName" : "BscIT",
                "duration" : "3 Year"
        },
        "address" : {
                "city" : "Mumbai",
                "State" : "MH",
                "country" : "india"
        }
}
>
```

DELETE DOCUMENT -



```
> db.student.remove({"regNo":"123456"})
WriteResult({ "nRemoved" : 1 })
```

PRACTICAL NO:02

SIMPLE QUERIES WITH MONGODB

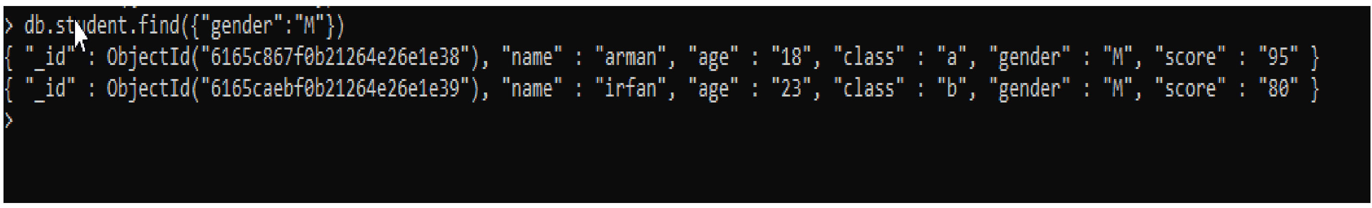
1) Selector – Syntax :

db.collection_name.find({"Key":"Value"})

Source code:

Use tyit db.student.find({"Gender":"M"})

OUTPUT :



```
> db.student.find({"gender":"M"})
{ "_id" : ObjectId("6165c867f0b21264e26e1e38"), "name" : "arman", "age" : "18", "class" : "a", "gender" : "M", "score" : "95" }
{ "_id" : ObjectId("6165caebf0b21264e26e1e39"), "name" : "irfan", "age" : "23", "class" : "b", "gender" : "M", "score" : "80" }
>
```

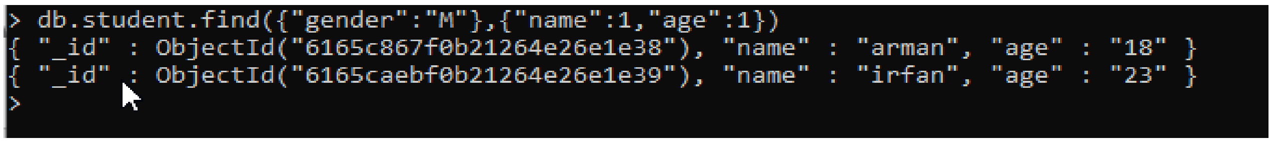
2) Projector - Syntax:

db.collection_name.find({"Key":"Value"},
{"Key":Value,"Key":Value})

Source code:

Use MYBase db.student.find({"Gender":"M"},{"Name":1,"Age":1})

Output:



```
> db.student.find({"gender":"M"},{"name":1,"age":1})
{ "_id" : ObjectId("6165c867f0b21264e26e1e38"), "name" : "arman", "age" : "18" }
{ "_id" : ObjectId("6165caebf0b21264e26e1e39"), "name" : "irfan", "age" : "23" }
>
```

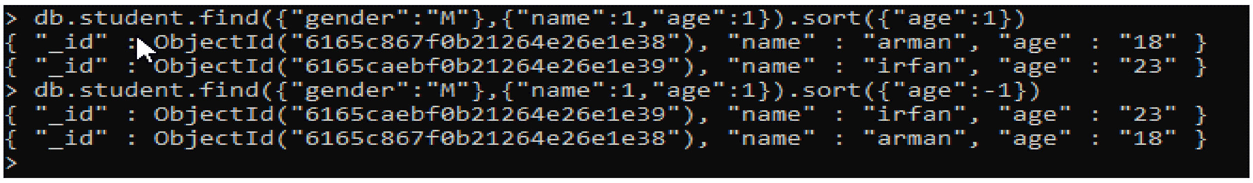
3) sort() - Syntax:

db.collection_name.find({"Key":"Value"},
{"Key":Value,"Key":Value}).sort({"Key":Value})

Source code:

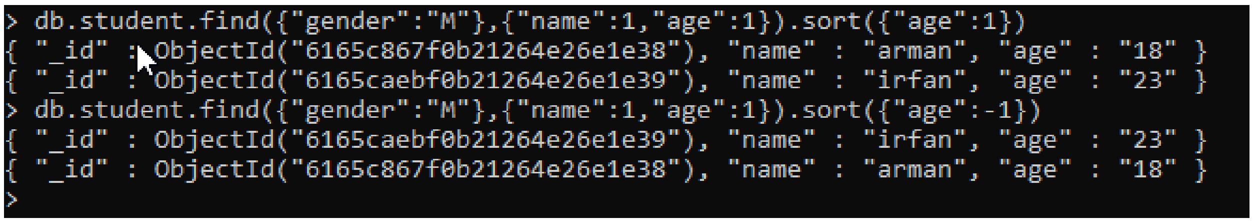
Use MYBase

db.stud.find({"Gender":"F"},{"Name":1,"Age":1}).sort({"Age":



```
> db.student.find({"gender":"M"},{"name":1,"age":1}).sort({"age":1})
{ "_id" : ObjectId("6165c867f0b21264e26e1e38"), "name" : "arman", "age" : "18" }
{ "_id" : ObjectId("6165caebf0b21264e26e1e39"), "name" : "irfan", "age" : "23" }
> db.student.find({"gender":"M"},{"name":1,"age":1}).sort({"age":-1})
{ "_id" : ObjectId("6165caebf0b21264e26e1e39"), "name" : "irfan", "age" : "23" }
{ "_id" : ObjectId("6165c867f0b21264e26e1e38"), "name" : "arman", "age" : "18" }
>
```

OUTPUT:



```
> db.student.find({"gender":"M"},{"name":1,"age":1}).sort({"age":1})
{ "_id" : ObjectId("6165c867f0b21264e26e1e38"), "name" : "arman", "age" : "18" }
{ "_id" : ObjectId("6165caebf0b21264e26e1e39"), "name" : "irfan", "age" : "23" }
> db.student.find({"gender":"M"},{"name":1,"age":1}).sort({"age":-1})
{ "_id" : ObjectId("6165caebf0b21264e26e1e39"), "name" : "irfan", "age" : "23" }
{ "_id" : ObjectId("6165c867f0b21264e26e1e38"), "name" : "arman", "age" : "18" }
>
```

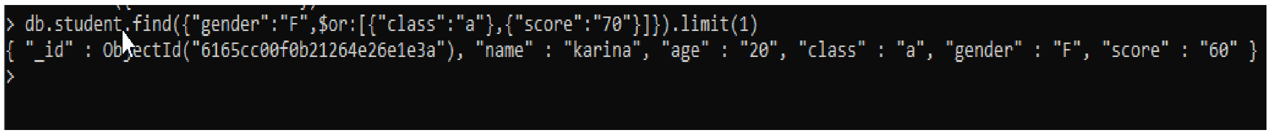
4) Limit() - Syntax:

db.collection_name.find({"Key":"Value",$or:[{"Key":"Value"},
{"Key":" Value"}]}).limit(Value)

Source code:

Use tyit db.stud.find({"gender":"F",$or:[{"class":"a"},{"score":"70"}
]}).limit(2)

Output:



```
> db.student.find({"gender":"F",$or:[{"class":"a"},{"score":"70"}]}).limit(1)
{ "_id" : ObjectId("6165cc00f0b21264e26e1e3a"), "name" : "karina", "age" : "20", "class" : "a", "gender" : "F", "score" : "60" }
>
```

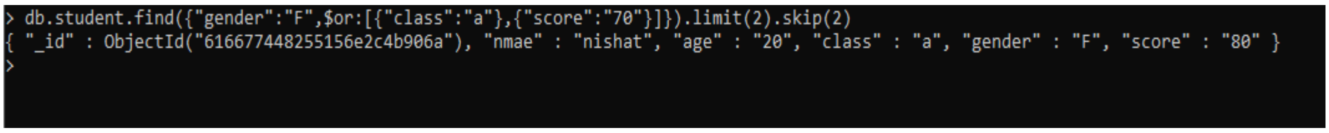
5) Skip() - Syntax:

db.collection_name.find({"Key":"Value",$or:[{"Key":"Value"},
{"Key":" Value"}]}).limit(Value).skip(Value)

Source code:

Use tyit db.stud.find({"Gender":"F",$or:[{"class":"a"},
{"score":"70"}]}).limit(2).skip(2)

Output:



```
> db.student.find({"gender":"F",$or:[{"class":"a"},{"score":"70"}]}).limit(2).skip(2)
{ "_id" : ObjectId("616677448255156e2c4b906a"), "nmae" : "nishat", "age" : "20", "class" : "a", "gender" : "F", "score" : "80" }
>
```

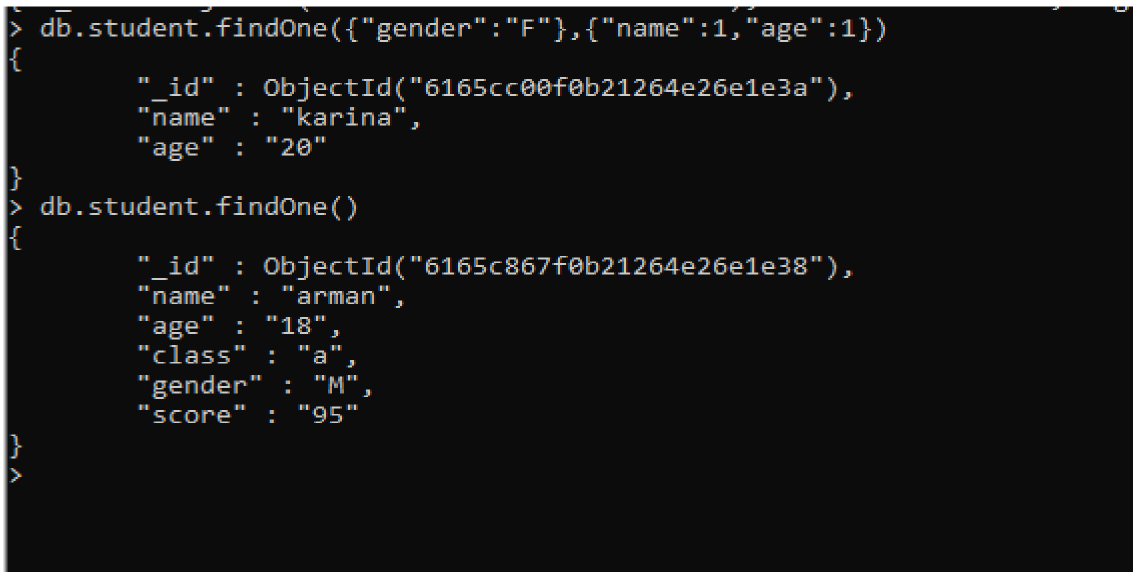
6) Findone() - Syntax:

db.collection_name.findOne({"Key":"Value"},
{"Key":Value,"Key":Value}) db.collection_name.findOne()

Source code:

Use tyit

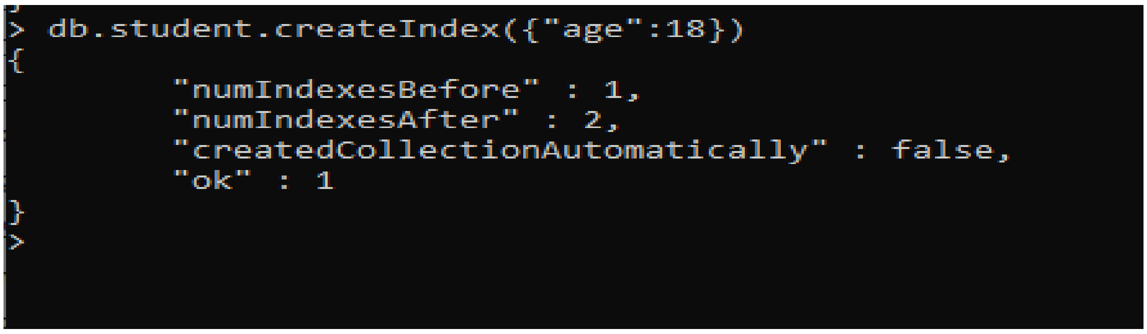
db.stud.findOne({"gender":"F"}, {"Name":1,"Age":1}) db.stud.findOne()
Output:



```
> db.student.findOne({"gender":"F"},{"name":1,"age":1})
{
        "_id" : ObjectId("6165cc00f0b21264e26e1e3a"),
        "name" : "karina",
        "age" : "20"
}
> db.student.findOne()
{
        "_id" : ObjectId("6165c867f0b21264e26e1e38"),
        "name" : "arman",
        "age" : "18",
        "class" : "a",
        "gender" : "M",
        "score" : "95"
}
>
```

7) ensureIndex Syntax:

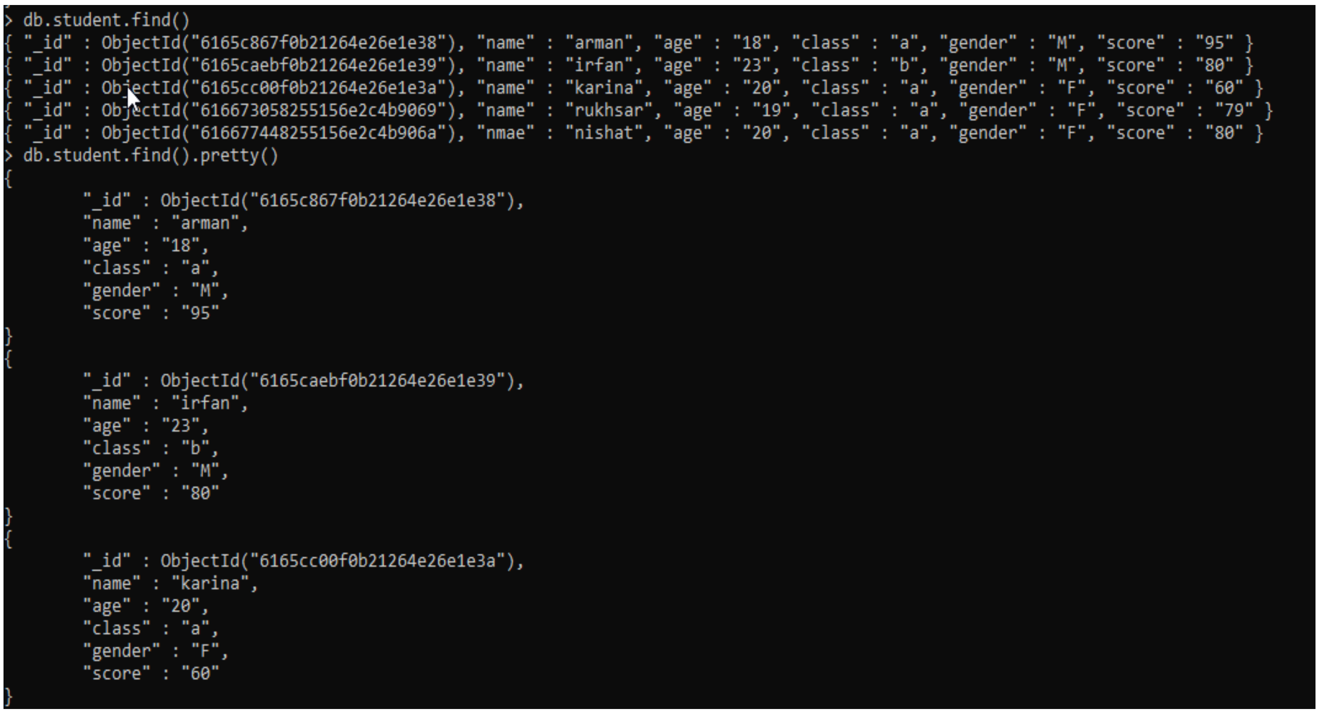
db.collection_name.ensureIndex({"Key":"Value"})
Source Code:
Use mydatabase Show collections
db.example.ensureIndex({“age”:26})
Output:



```
> db.student.createIndex({"age":18})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
>
```

8) Pretty() - Syntax :

db.collection_name.find().pretty()
Source Code:
use tyit
show collections
db.user1.find() db.user1.find.pretty()
Output:



```
> db.student.find()
{ "_id" : ObjectId("6165c867f0b21264e26e1e38"), "name" : "arman", "age" : "18", "class" : "a", "gender" : "M", "score" : "95" }
{ "_id" : ObjectId("6165caebf0b21264e26e1e39"), "name" : "irfan", "age" : "23", "class" : "b", "gender" : "M", "score" : "80" }
{ "_id" : ObjectId("6165cc00f0b21264e26e1e3a"), "name" : "karina", "age" : "20", "class" : "a", "gender" : "F", "score" : "60" }
{ "_id" : ObjectId("616673058255156e2c4b9069"), "name" : "rukhsar", "age" : "19", "class" : "a", "gender" : "F", "score" : "79" }
{ "_id" : ObjectId("616677448255156e2c4b906a"), "nmae" : "nishat", "age" : "20", "class" : "a", "gender" : "F", "score" : "80" }
> db.student.find().pretty()
{
        "_id" : ObjectId("6165c867f0b21264e26e1e38"),
        "name" : "arman",
        "age" : "18",
        "class" : "a",
        "gender" : "M",
        "score" : "95"
}
{
        "_id" : ObjectId("6165caebf0b21264e26e1e39"),
        "name" : "irfan",
        "age" : "23",
        "class" : "b",
        "gender" : "M",
        "score" : "80"
}
{
        "_id" : ObjectId("6165cc00f0b21264e26e1e3a"),
        "name" : "karina",
        "age" : "20",
        "class" : "a",
        "gender" : "F",
        "score" : "60"
}
```

9) Conditional Operators - Syntax:

$lt and $lte

db.collection_name.find({"Key":{"$lt":Value}})

db.collection_name.find({"Key":{"$lte":Value}})

$gt and $gte

db.collection_name.find({"Key":{"$gt":Value}})

db.collection_name.find({"Key":{"$gte":Value}})

$in and $nin

db.collection_name.find({"Key":{"$in":["Value","Value"]}})

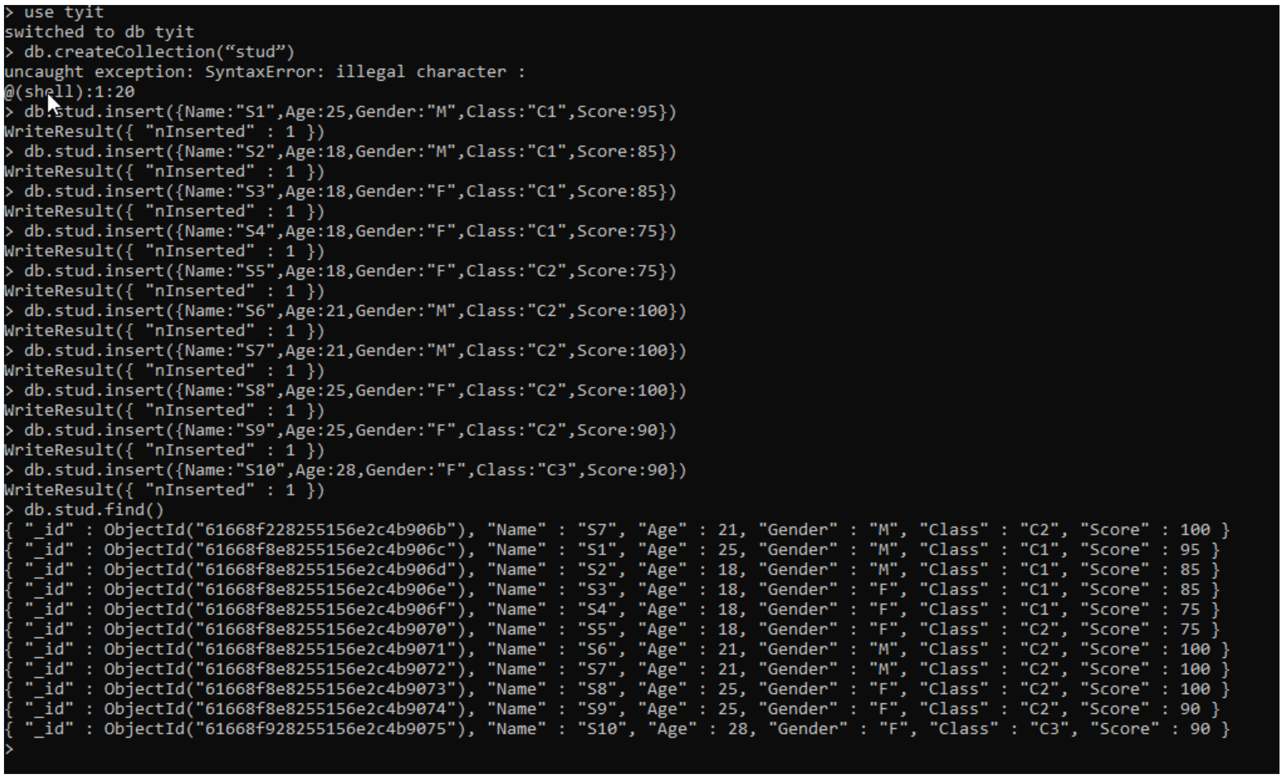
db.collection_name.find({"Key":{"$nin":["Value","Value"]}})

Source code:

```
use tyit
db.createCollection("stud")
db.stud.insert({Name:"S1",Age:25,Gender:"M",Class:"C1",Sc ore:95})
db.stud.insert({Name:"S2",Age:18,Gender:"M",Class:"C1",Sc ore:85})
db.stud.insert({Name:"S3",Age:18,Gender:"F",Class:"C1",Sco re:85})
db.stud.insert({Name:"S4",Age:18,Gender:"F",Class:"C1",Sco re:75})
db.stud.insert({Name:"S5",Age:18,Gender:"F",Class:"C2",Sco re:75})
db.stud.insert({Name:"S6",Age:21,Gender:"M",Class:"C2",Sc ore:100})
db.stud.insert({Name:"S7",Age:21,Gender:"M",Class:"C2",Sc ore:100})
db.stud.insert({Name:"S8",Age:25,Gender:"F",Class:"C2",Sco re:100})
db.stud.insert({Name:"S9",Age:25,Gender:"F",Class:"C2",Sco re:90})
db.stud.insert({Name:"S10",Age:28,Gender:"F",Class:"C3",Sc ore:90})
db.stud.find()
$lt and $lte db.stud.find({"Age":{"$lt":25}})
db.stud.find({"Age":{"$lte":25}})
```

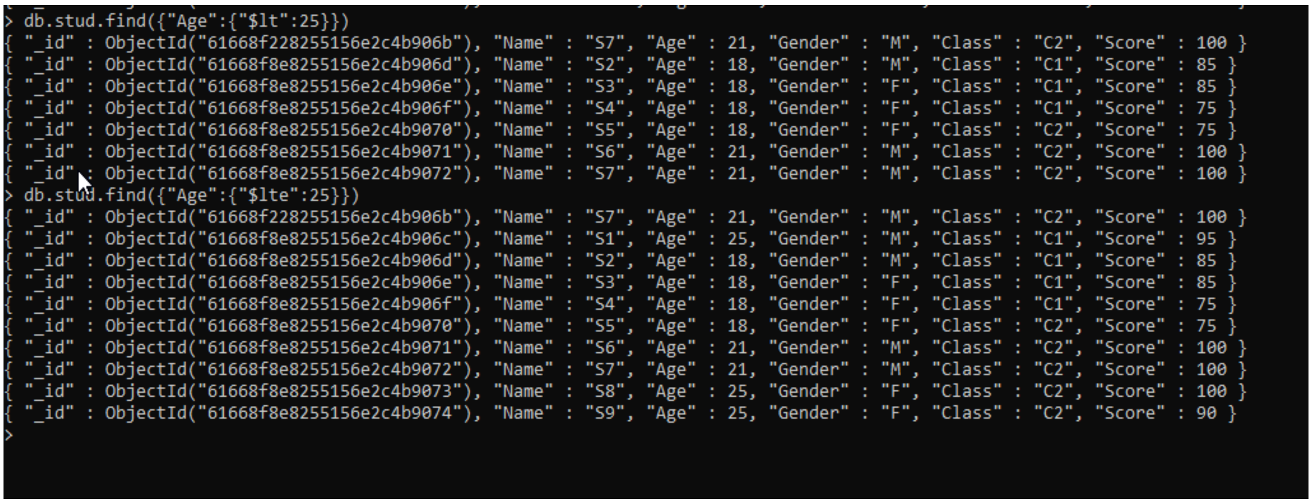
$gt and $gte db.stud.find({"Age":{"$gt":25}})
db.stud.find({"Age":{"$gte":25}})
$in and $nin
db.stud.find({"Class":{"$in":["C1","C2"]}})
db.stud.find({"Class":{"$nin":["C1","C2"]}})

Output:



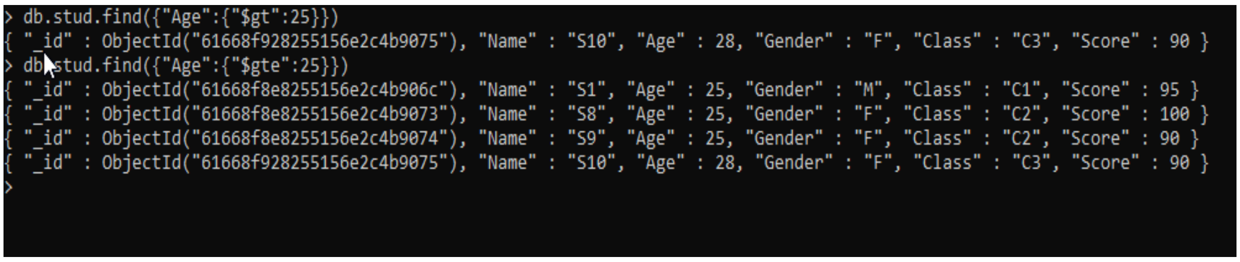
```
> use tyit
switched to db tyit
> db.createCollection(“stud”)
uncaught exception: SyntaxError: illegal character :
@(shell):1:20
> db.stud.insert({Name:"S1",Age:25,Gender:"M",Class:"C1",Score:95})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S2",Age:18,Gender:"M",Class:"C1",Score:85})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S3",Age:18,Gender:"F",Class:"C1",Score:85})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S4",Age:18,Gender:"F",Class:"C1",Score:75})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S5",Age:18,Gender:"F",Class:"C2",Score:75})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S6",Age:21,Gender:"M",Class:"C2",Score:100})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S7",Age:21,Gender:"M",Class:"C2",Score:100})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S8",Age:25,Gender:"F",Class:"C2",Score:100})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S9",Age:25,Gender:"F",Class:"C2",Score:90})
WriteResult({ "nInserted" : 1 })
> db.stud.insert({Name:"S10",Age:28,Gender:"F",Class:"C3",Score:90})
WriteResult({ "nInserted" : 1 })
> db.stud.find()
{ "_id" : ObjectId("61668f228255156e2c4b906b"), "Name" : "S7", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906c"), "Name" : "S1", "Age" : 25, "Gender" : "M", "Class" : "C1", "Score" : 95 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906d"), "Name" : "S2", "Age" : 18, "Gender" : "M", "Class" : "C1", "Score" : 85 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906e"), "Name" : "S3", "Age" : 18, "Gender" : "F", "Class" : "C1", "Score" : 85 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906f"), "Name" : "S4", "Age" : 18, "Gender" : "F", "Class" : "C1", "Score" : 75 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9070"), "Name" : "S5", "Age" : 18, "Gender" : "F", "Class" : "C2", "Score" : 75 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9071"), "Name" : "S6", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9072"), "Name" : "S7", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9073"), "Name" : "S8", "Age" : 25, "Gender" : "F", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9074"), "Name" : "S9", "Age" : 25, "Gender" : "F", "Class" : "C2", "Score" : 90 }
{ "_id" : ObjectId("61668f928255156e2c4b9075"), "Name" : "S10", "Age" : 28, "Gender" : "F", "Class" : "C3", "Score" : 90 }
>
```

$lt and $lte :



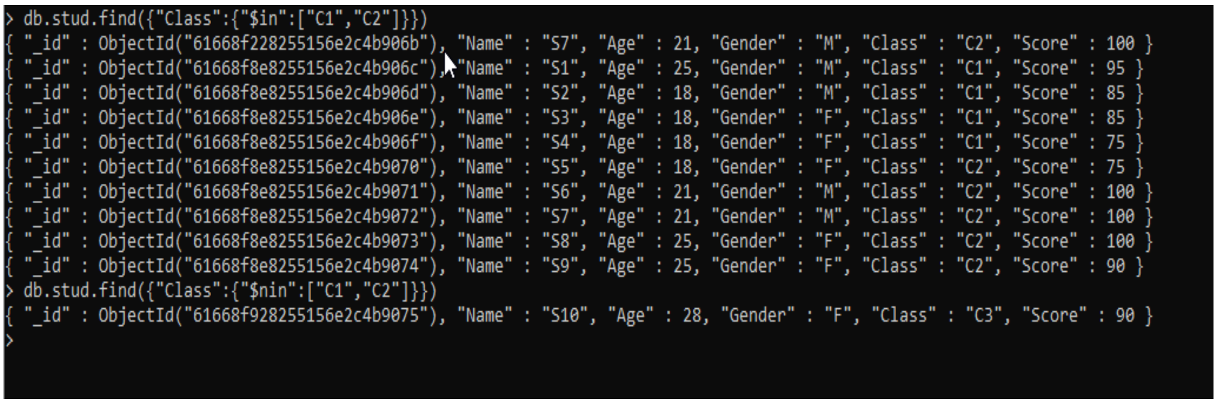
```
> db.stud.find({"Age":{"$lt":25}})
{ "_id" : ObjectId("61668f228255156e2c4b906b"), "Name" : "S7", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906d"), "Name" : "S2", "Age" : 18, "Gender" : "M", "Class" : "C1", "Score" : 85 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906e"), "Name" : "S3", "Age" : 18, "Gender" : "F", "Class" : "C1", "Score" : 85 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906f"), "Name" : "S4", "Age" : 18, "Gender" : "F", "Class" : "C1", "Score" : 75 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9070"), "Name" : "S5", "Age" : 18, "Gender" : "F", "Class" : "C2", "Score" : 75 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9071"), "Name" : "S6", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9072"), "Name" : "S7", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
> db.stud.find({"Age":{"$lte":25}})
{ "_id" : ObjectId("61668f228255156e2c4b906b"), "Name" : "S7", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906c"), "Name" : "S1", "Age" : 25, "Gender" : "M", "Class" : "C1", "Score" : 95 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906d"), "Name" : "S2", "Age" : 18, "Gender" : "M", "Class" : "C1", "Score" : 85 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906e"), "Name" : "S3", "Age" : 18, "Gender" : "F", "Class" : "C1", "Score" : 85 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906f"), "Name" : "S4", "Age" : 18, "Gender" : "F", "Class" : "C1", "Score" : 75 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9070"), "Name" : "S5", "Age" : 18, "Gender" : "F", "Class" : "C2", "Score" : 75 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9071"), "Name" : "S6", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9072"), "Name" : "S7", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9073"), "Name" : "S8", "Age" : 25, "Gender" : "F", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9074"), "Name" : "S9", "Age" : 25, "Gender" : "F", "Class" : "C2", "Score" : 90 }
>
```

$gt and $gte



```
> db.stud.find({"Age":{"$gt":25}})
{ "_id" : ObjectId("61668f928255156e2c4b9075"), "Name" : "S10", "Age" : 28, "Gender" : "F", "Class" : "C3", "Score" : 90 }
> db.stud.find({"Age":{"$gte":25}})
{ "_id" : ObjectId("61668f8e8255156e2c4b906c"), "Name" : "S1", "Age" : 25, "Gender" : "M", "Class" : "C1", "Score" : 95 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9073"), "Name" : "S8", "Age" : 25, "Gender" : "F", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9074"), "Name" : "S9", "Age" : 25, "Gender" : "F", "Class" : "C2", "Score" : 90 }
{ "_id" : ObjectId("61668f928255156e2c4b9075"), "Name" : "S10", "Age" : 28, "Gender" : "F", "Class" : "C3", "Score" : 90 }
>
```

$in and $nin



```
> db.stud.find({"Class":{"$in":["C1","C2"]}})
{ "_id" : ObjectId("61668f228255156e2c4b906b"), "Name" : "S7", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906c"), "Name" : "S1", "Age" : 25, "Gender" : "M", "Class" : "C1", "Score" : 95 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906d"), "Name" : "S2", "Age" : 18, "Gender" : "M", "Class" : "C1", "Score" : 85 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906e"), "Name" : "S3", "Age" : 18, "Gender" : "F", "Class" : "C1", "Score" : 85 }
{ "_id" : ObjectId("61668f8e8255156e2c4b906f"), "Name" : "S4", "Age" : 18, "Gender" : "F", "Class" : "C1", "Score" : 75 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9070"), "Name" : "S5", "Age" : 18, "Gender" : "F", "Class" : "C2", "Score" : 75 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9071"), "Name" : "S6", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9072"), "Name" : "S7", "Age" : 21, "Gender" : "M", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9073"), "Name" : "S8", "Age" : 25, "Gender" : "F", "Class" : "C2", "Score" : 100 }
{ "_id" : ObjectId("61668f8e8255156e2c4b9074"), "Name" : "S9", "Age" : 25, "Gender" : "F", "Class" : "C2", "Score" : 90 }
> db.stud.find({"Class":{"$nin":["C1","C2"]}})
{ "_id" : ObjectId("61668f928255156e2c4b9075"), "Name" : "S10", "Age" : 28, "Gender" : "F", "Class" : "C3", "Score" : 90 }
>
```

PRACTICAL NO:03

IMPLEMENTING AGGREGATION

a. Write a MongoDB query to use sum, avg, min, and max expression.
b. Write a MongoDB query to use push and addToSet expression.
c. Write a MongoDB query to use first and last expression.

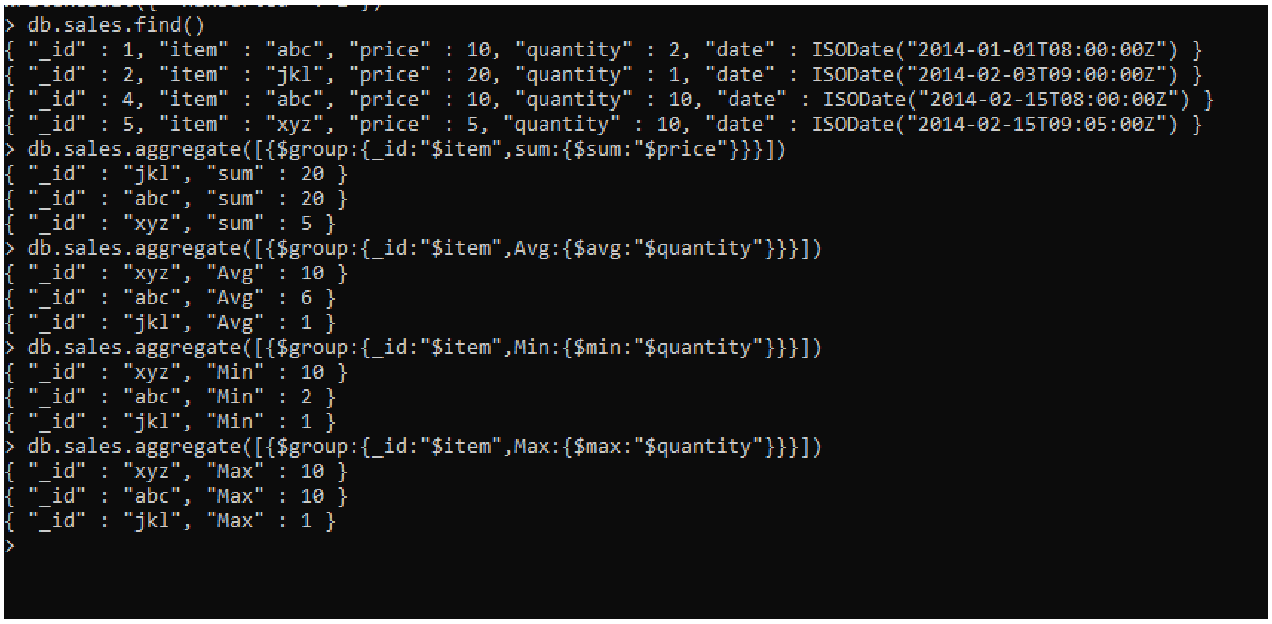
3) a - Write a MongoDB query to use sum, avg, min, and max expression.
Syntax:

- Sum:
  {$sum: [<expression1>, <expression>...]}
- Avg:
  {$avg: [<expression1>...]}
- Min:
  {$min: [<expression1>...]}
- Max:
  {$max: [<expression1>...]}

Source code:

```
use minimum
db.createCollection("Sales")
db.Sales.insert({ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2,
"date" : ISODate("2014-01-01T08:00:00Z") }
db.Sales.insert { "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1,
"date" : ISODate("2014-02-03T09:00:00Z") }
db.Sales.insert { "_id" : 3, "item" : "xyz", "price" : 5, "quantity"
: 5,
"date" : ISODate("2014-02-03T09:05:00Z") }
db.Sales.insert { "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10,
"date" : ISODate("2014-02-15T08:00:00Z") }
db.Sales.insert { "_id" : 5, "item" : "xyz", "price" : 5, "quantity"
: 10,
"date" : ISODate("2014-02-15T09:05:00Z") }
db.sales.aggregate([{$group:{_id:"$item",sum:{$sum:"$price"
}}}])
db.sales.aggregate([{$group:{_id:"$item",Avg:{$avg:"$quantit y"}}}])
db.sales.aggregate([{$group:{_id:"$item",Min:{$min:"$quanti ty"}}}])
db.sales.aggregate([{$group:{_id:"$item",Max:{$max:"$quant ity"}}}])
```

OUTPUT :



```
> db.sales.find()
{ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-01-01T08:00:00Z") }
{ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-02-03T09:00:00Z") }
{ "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-02-15T08:00:00Z") }
{ "_id" : 5, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-02-15T09:05:00Z") }
> db.sales.aggregate([{$group:{_id:"$item",sum:{$sum:"$price"}}}])
{ "_id" : "jkl", "sum" : 20 }
{ "_id" : "abc", "sum" : 20 }
{ "_id" : "xyz", "sum" : 5 }
> db.sales.aggregate([{$group:{_id:"$item",Avg:{$avg:"$quantity"}}}])
{ "_id" : "xyz", "Avg" : 10 }
{ "_id" : "abc", "Avg" : 6 }
{ "_id" : "jkl", "Avg" : 1 }
> db.sales.aggregate([{$group:{_id:"$item",Min:{$min:"$quantity"}}}])
{ "_id" : "xyz", "Min" : 10 }
{ "_id" : "abc", "Min" : 2 }
{ "_id" : "jkl", "Min" : 1 }
> db.sales.aggregate([{$group:{_id:"$item",Max:{$max:"$quantity"}}}])
{ "_id" : "xyz", "Max" : 10 }
{ "_id" : "abc", "Max" : 10 }
{ "_id" : "jkl", "Max" : 1 }
>
```

3.b.Write a MongoDB query to use push and addToSet expression.

Syntax:

Push

{$push:<expression>} addToSet

{$addToSet:<expression>}

Source code:

use minimum

db.createCollection(“sales”)

db.sales.insert({ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2,

"date" : ISODate("2014-01-01T08:00:00Z") }

db.sales.insert { "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1,

"date" : ISODate("2014-02-03T09:00:00Z") }

db.sales.insert { "_id" : 3, "item" : "xyz", "price" : 5, "quantity"

: 5,

"date" : ISODate("2014-02-03T09:05:00Z") }

db.sales.insert { "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10,

"date" : ISODate("2014-02-15T08:00:00Z") }

db.sales.insert { "_id" : 5, "item" : "xyz", "price" : 5, "quantity"

: 10,

"date" : ISODate("2014-02-15T09:05:00Z") }

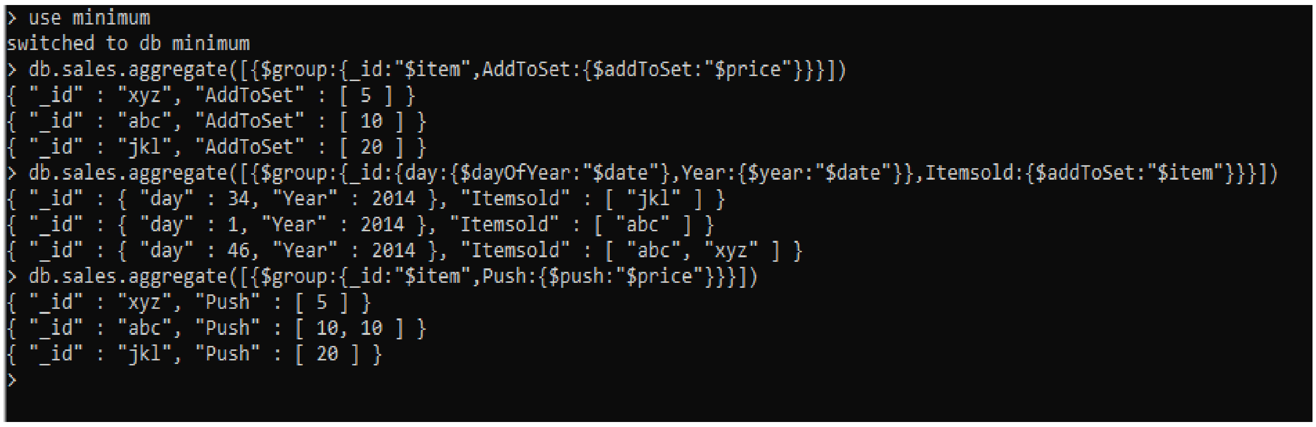
db.sales.aggregate([{$group:{_id:"$item",AddToSet:{$addToS et:"$pri ce"}}}])

db.sales.aggregate([{$group:{_id:{day:{$dayOfYear:"$date"},Y ear:{$y

ear:"$date"}},Itemsold:{$addToSet:"$item"}}}])

db.sales.aggregate([{$group:{_id:"$item",Push:{$push:"$price "}}}])

OUTPUT :



```
> use minimum
switched to db minimum
> db.sales.aggregate([{$group:{_id:"$item",AddToSet:{$addToSet:"$price"}}}])
{ "_id" : "xyz", "AddToSet" : [ 5 ] }
{ "_id" : "abc", "AddToSet" : [ 10 ] }
{ "_id" : "jkl", "AddToSet" : [ 20 ] }
> db.sales.aggregate([{$group:{_id:{day:{$dayOfYear:"$date"},Year:{$year:"$date"}},Itemsold:{$addToSet:"$item"}}}])
{ "_id" : { "day" : 34, "Year" : 2014 }, "Itemsold" : [ "jkl" ] }
{ "_id" : { "day" : 1, "Year" : 2014 }, "Itemsold" : [ "abc" ] }
{ "_id" : { "day" : 46, "Year" : 2014 }, "Itemsold" : [ "abc", "xyz" ] }
> db.sales.aggregate([{$group:{_id:"$item",Push:{$push:"$price"}}}])
{ "_id" : "xyz", "Push" : [ 5 ] }
{ "_id" : "abc", "Push" : [ 10, 10 ] }
{ "_id" : "jkl", "Push" : [ 20 ] }
>
```

3.c. Write a MongoDB query to use first and last expression. Syntax:

First

{$first:<expression>}

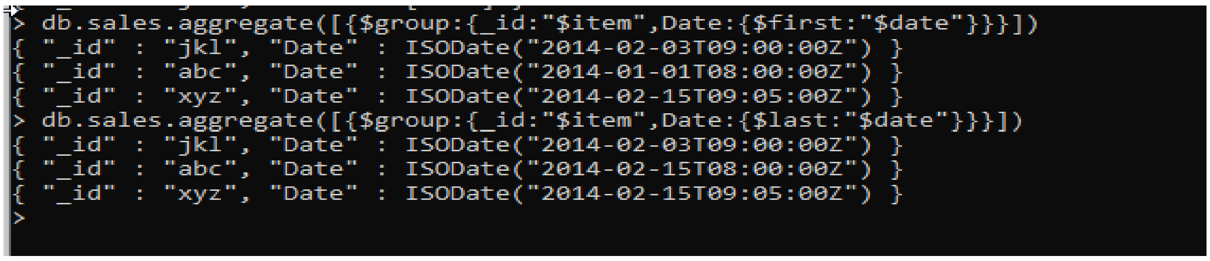
Last

{$last:<expression>}

Source code:

use minimum
db.createCollection(“Sales”)
db.Sales.insert({ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2,
"date" : ISODate("2014-01-01T08:00:00Z") }
db.Sales.insert { "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1,
"date" : ISODate("2014-02-03T09:00:00Z") }
db.Sales.insert { "_id" : 3, "item" : "xyz", "price" : 5, "quantity"
: 5,
"date" : ISODate("2014-02-03T09:05:00Z") }
db.Sales.insert { "_id" : 4, "item" : "abc", "price" : 10, "quantity" : 10,
"date" : ISODate("2014-02-15T08:00:00Z") }
db.Sales.insert { "_id" : 5, "item" : "xyz", "price" : 5, "quantity"
: 10,
"date" : ISODate("2014-02-15T09:05:00Z") }
db.sales.aggregate([{$group:{_id:"$item",Date:{$first:"$date"
}}}])
db.sales.aggregate([{$group:{_id:"$item",Date:{$last:"$date"}
}}])

OUTPUT :



```
> db.sales.aggregate([{$group:{_id:"$item",Date:{$first:"$date"}}}])
{ "_id" : "jkl", "Date" : ISODate("2014-02-03T09:00:00Z") }
{ "_id" : "abc", "Date" : ISODate("2014-01-01T08:00:00Z") }
{ "_id" : "xyz", "Date" : ISODate("2014-02-15T09:05:00Z") }
> db.sales.aggregate([{$group:{_id:"$item",Date:{$last:"$date"}}}])
{ "_id" : "jkl", "Date" : ISODate("2014-02-03T09:00:00Z") }
{ "_id" : "abc", "Date" : ISODate("2014-02-15T08:00:00Z") }
{ "_id" : "xyz", "Date" : ISODate("2014-02-15T09:05:00Z") }
>
```

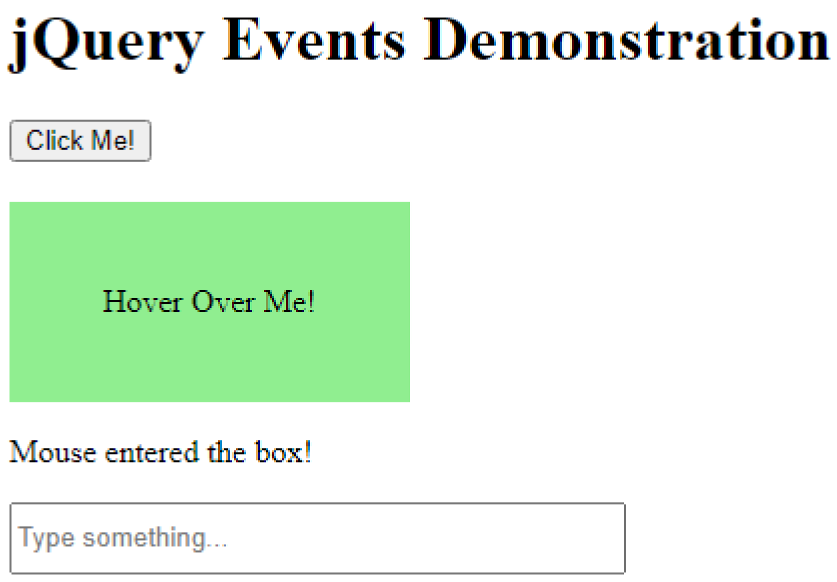
PRACTICAL NO:04

Aim: Write a program to demonstrate jQuery Events

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>jQuery Events Demo</title>
<!-- jQuery CDN -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<style>
#hoverBox { width: 200px; height: 100px;
background-color: lightblue; text-align: center;
line-height: 100px; margin-top: 20px;
}
#textBox { width: 300px; height: 30px;
}
</style>
</head>
<body>

<h1>jQuery Events Demonstration</h1>

<!-- Button to demonstrate click event -->
<button id="clickButton">Click Me!</button>
<p id="clickResult"></p>

<!-- Hover box to demonstrate mouseenter and mouseleave events -->
<div id="hoverBox">Hover Over Me!</div>
<p id="hoverResult"></p>

<!-- Input box to demonstrate keypress event -->
<input type="text" id="textBox" placeholder="Type something...">
<p id="keypressResult"></p>

<script>
$(document).ready(function(){
// Click event

$("#clickButton").click(function(){
$("#clickResult").text("Button was clicked!");
});

// Mouseenter and Mouseleave events
$("#hoverBox").mouseenter(function(){
```

```
$("#hoverResult").text("Mouse entered the box!");
$(this).css("background-color", "lightgreen");
});

$("#hoverBox").mouseleave(function(){
$("#hoverResult").text("Mouse left the box!");
$(this).css("background-color", "lightblue");
});

// Keypress event
$("#textBox").keypress(function(event){
$("#keypressResult").text("You pressed: " + String.fromCharCode(event.which));
});
});
</script>

</body>
</html>
```

Output:


jQuery Events Demonstration

Mouse entered the box!

PRACTICAL NO:05

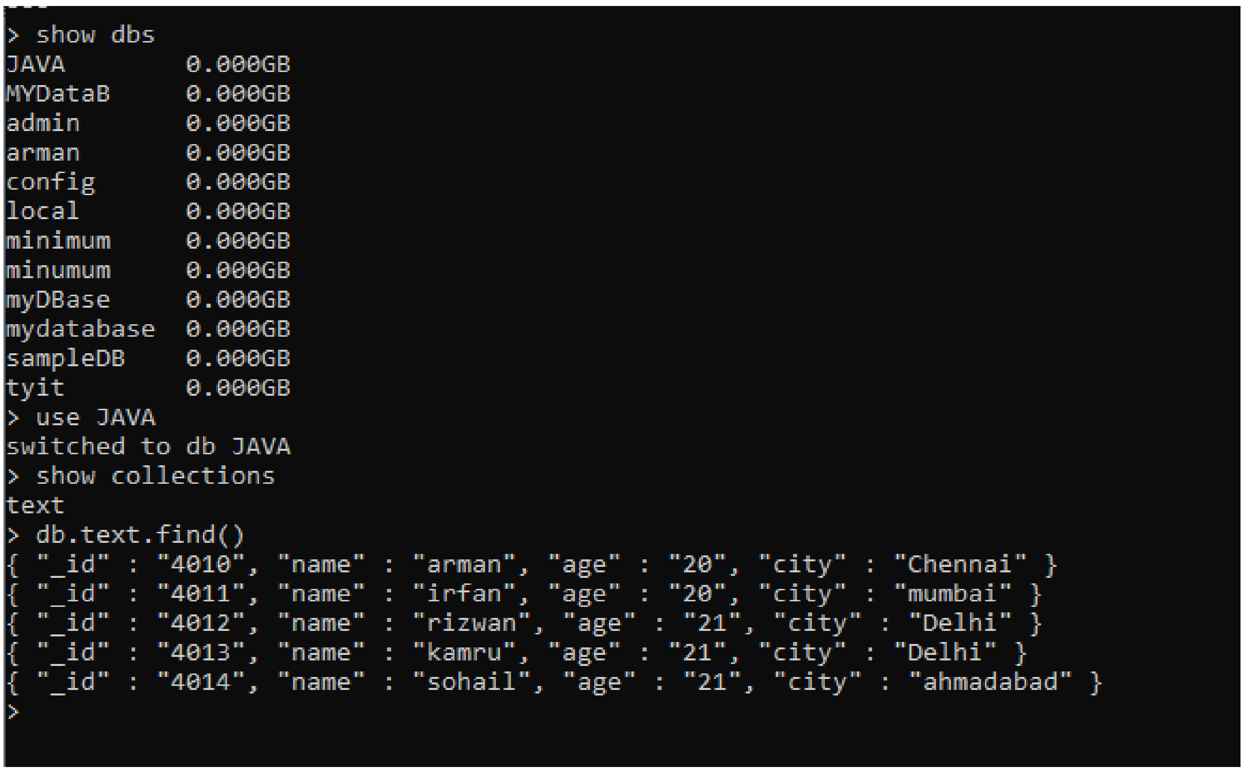
Java and MongoDB

1) Inserting Data -

Source Code :

```
package javamongodb; import com.mongodb.DB; import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable; import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase; import com.mongodb.client.model.Filters;
import java.util.Iterator; import org.bson.Document;
public static void main(String[] args) { try{
MongoClient mongoclient=new MongoClient("localhost",27017);
MongoDatabase db=mongoclient.getDatabase("JAVA"); System.out.println("Connected to
Database"); Document document = new Document();
document.append("_id","4010"); document.append("name","arman");
document.append("age", "20");
document.append("city", "Chennai"); document.append("_id","4011");
document.append("name","irfan");
document.append("age", "20"); document.append("city", "mumbai");
document.append("_id","4012"); document.append("name","rizwan");
document.append("age", "21"); document.append("city", "Delhi");
document.append("_id","4013"); document.append("name","kamru");
document.append("age", "22"); document.append("city", "Delhi");
document.append("_id","4014");
document.append("name","sohail"); document.append("age", "21");
document.append("city", "Banglore");
db.getCollection("text").insertOne(document);
System.out.println("Data Inserted Successfully !!!"); FindIterable
iterDoc=db.getCollection("text").find();
int i=1;
Iterator it=iterDoc.iterator();
while(it.hasNext()){ System.out.println(it.next()); i++;
}
}catch(Exception e)
{
System.out.println(e);
}
}
}
```

OUTPUT :



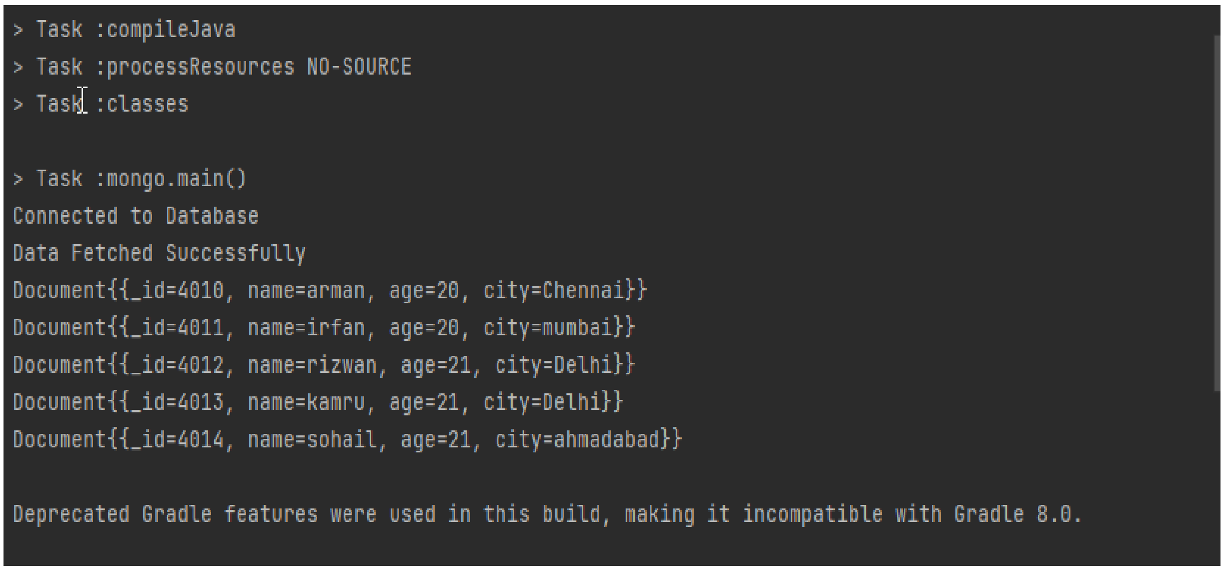
```
> show dbs
JAVA         0.000GB
MYDataB      0.000GB
admin        0.000GB
arman        0.000GB
config       0.000GB
local        0.000GB
minimum      0.000GB
minumum      0.000GB
myDBase      0.000GB
mydatabase   0.000GB
sampleDB     0.000GB
tyit         0.000GB
> use JAVA
switched to db JAVA
> show collections
text
> db.text.find()
{ "_id" : "4010", "name" : "arman", "age" : "20", "city" : "Chennai" }
{ "_id" : "4011", "name" : "irfan", "age" : "20", "city" : "mumbai" }
{ "_id" : "4012", "name" : "rizwan", "age" : "21", "city" : "Delhi" }
{ "_id" : "4013", "name" : "kamru", "age" : "21", "city" : "Delhi" }
{ "_id" : "4014", "name" : "sohail", "age" : "21", "city" : "ahmadabad" }
>
```

2) Retreive -

Source Code :

```
package javamongodb; import com.mongodb.DB; import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable; import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase; import com.mongodb.client.model.Filters;
import java.util.Iterator; import org.bson.Document;
public static void main(String[] args) {           try{
MongoClient mongoclient=new MongoClient("localhost",27017);
MongoDatabase db=mongoclient.getDatabase("JAVA");
System.out.println("Connected to Database"); FindIterable
iterDoc=db.getCollection("text").find();
int i=1;
Iterator it=iterDoc.iterator();  while(it.hasNext()){ System.out.println(it.next());
i++;      } }catch(Exception e)
{
System.out.println(e);
}
}
}
```

OUTPUT :

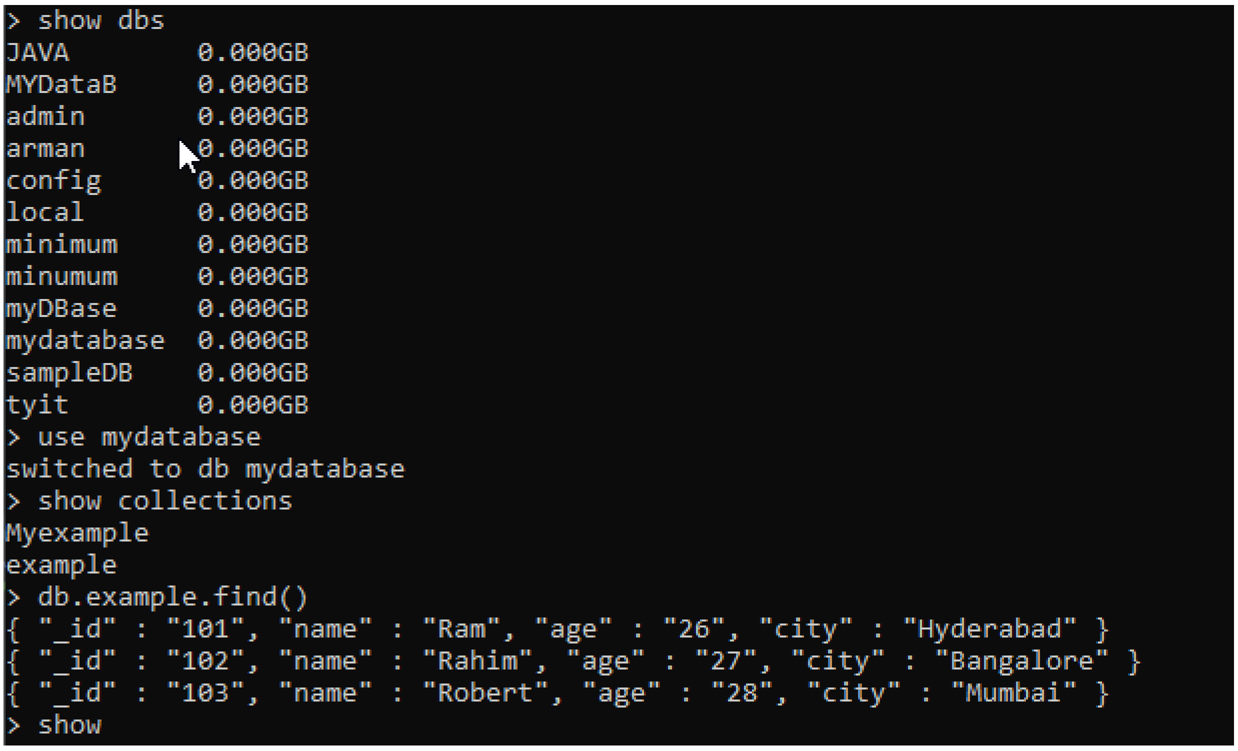


```
> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes

> Task :mongo.main()
Connected to Database
Data Fetched Successfully
Document{{_id=4010, name=arman, age=20, city=Chennai}}
Document{{_id=4011, name=irfan, age=20, city=mumbai}}
Document{{_id=4012, name=rizwan, age=21, city=Delhi}}
Document{{_id=4013, name=kamru, age=21, city=Delhi}}
Document{{_id=4014, name=sohail, age=21, city=ahmadabad}}

Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.
```

3)Update Source Code :

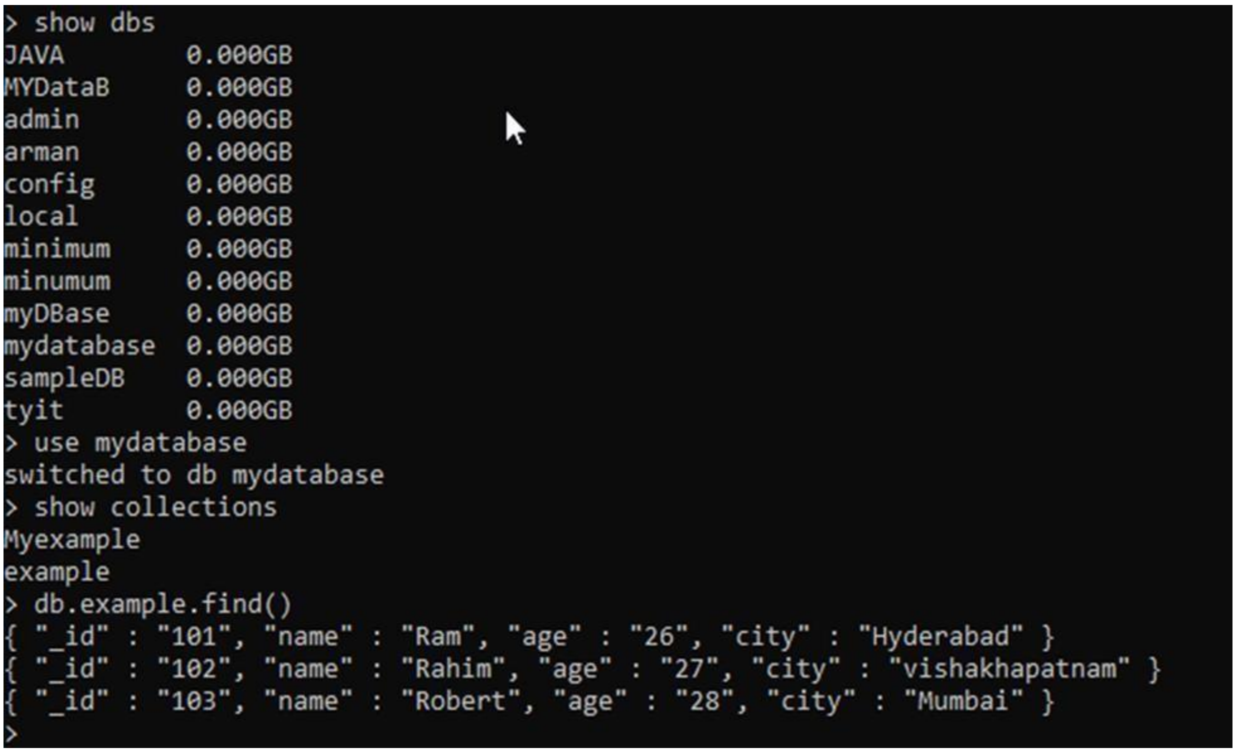
```
package javamongodb; import com.mongodb.DB; import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable; import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase; import com.mongodb.client.model.Filters;
import
com.mongodb.client.model.Updates; import java.util.Iterator; import org.bson.Document;
public class update {
public static void main(String[] args) {          try{
MongoClient mongoclient=new MongoClient("localhost",27017);
MongoDatabase db=mongoclient.getDatabase("mydatabase");
System.out.println("Connected to Database");

db.getCollection("example").updateOne(Filters.eq("_id","1 02" ), Updates.set("city",
"vishakapatnam"));
System.out.println("Data Updated Successfully !!!"); FindIterable
iterDoc=db.getCollection("example").find(); int i=1;
Iterator it=iterDoc.iterator(); while(it.hasNext()){ System.out.println(it.next());     i++;
}
}catch(Exception e)
{
System.out.println(e);
}
}
```

Before update :



```
> show dbs
JAVA         0.000GB
MYDataB      0.000GB
admin        0.000GB
arman        0.000GB
config       0.000GB
local        0.000GB
minimum      0.000GB
minumum      0.000GB
myDBase      0.000GB
mydatabase   0.000GB
sampleDB     0.000GB
tyit         0.000GB
> use mydatabase
switched to db mydatabase
> show collections
Myexample
example
> db.example.find()
{ "_id" : "101", "name" : "Ram", "age" : "26", "city" : "Hyderabad" }
{ "_id" : "102", "name" : "Rahim", "age" : "27", "city" : "Bangalore" }
{ "_id" : "103", "name" : "Robert", "age" : "28", "city" : "Mumbai" }
> show
```

After update :



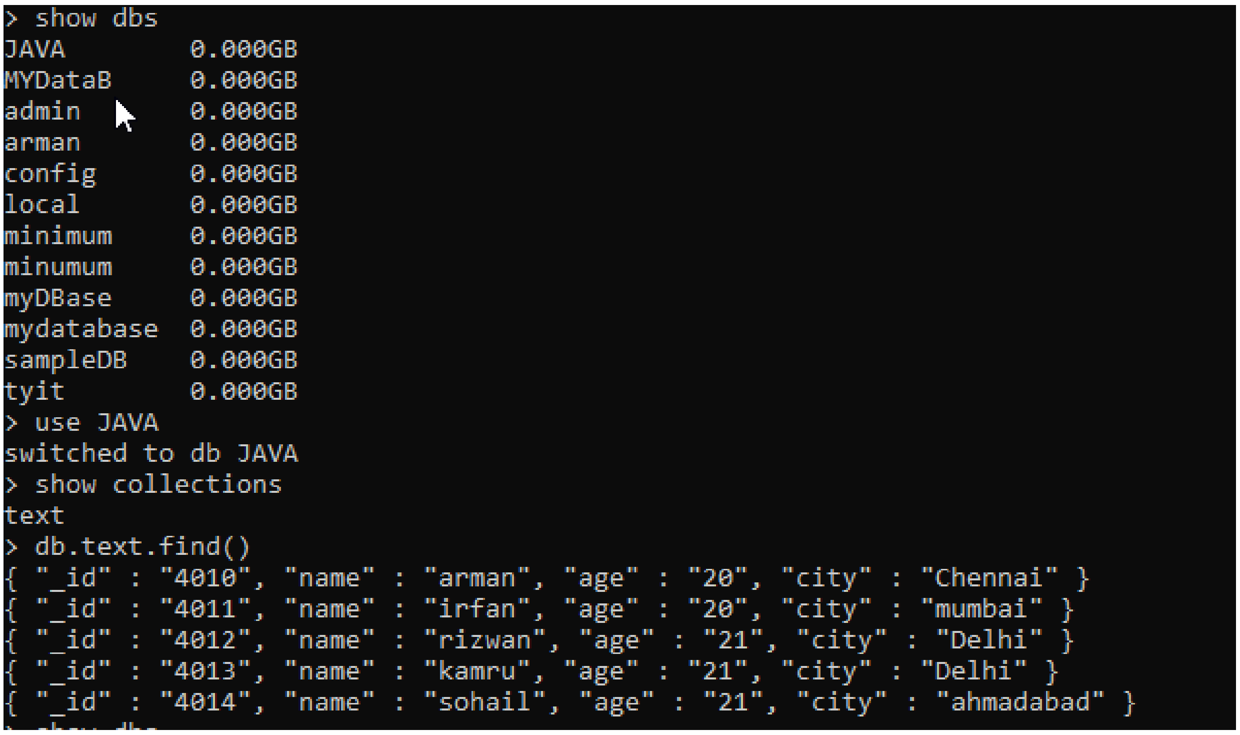
```
> show dbs
JAVA         0.000GB
MYDataB      0.000GB
admin        0.000GB
arman        0.000GB
config       0.000GB
local        0.000GB
minimum      0.000GB
minumum      0.000GB
myDBase      0.000GB
mydatabase   0.000GB
sampleDB     0.000GB
tyit         0.000GB
> use mydatabase
switched to db mydatabase
> show collections
Myexample
example
> db.example.find()
{ "_id" : "101", "name" : "Ram", "age" : "26", "city" : "Hyderabad" }
{ "_id" : "102", "name" : "Rahim", "age" : "27", "city" : "vishakhapatnam" }
{ "_id" : "103", "name" : "Robert", "age" : "28", "city" : "Mumbai" }
>
```

4) Delete Source Code :

```
package javamongodb; import com.mongodb.DB; import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable; import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase; import com.mongodb.client.model.Filters;
import java.util.Iterator; import org.bson.Document; public class delete {
public static void main(String[] arg)    try{
MongoClient mongoclient=new MongoClient("localhost",27017);
MongoDatabase db=mongoclient.getDatabase("mydatabase");
System.out.println("Connected to Database");

db.getCollection("example").deleteOne(Filters.eq("_id","4 009")
);
FindIterable iterDoc=db.getCollection("example").find(); int i=1;
Iterator it=iterDoc.iterator();  while(it.hasNext()){ System.out.println(it.next());      i++;
}
}catch(Exception e)
{
System.out.println(e);
}
}
}
```

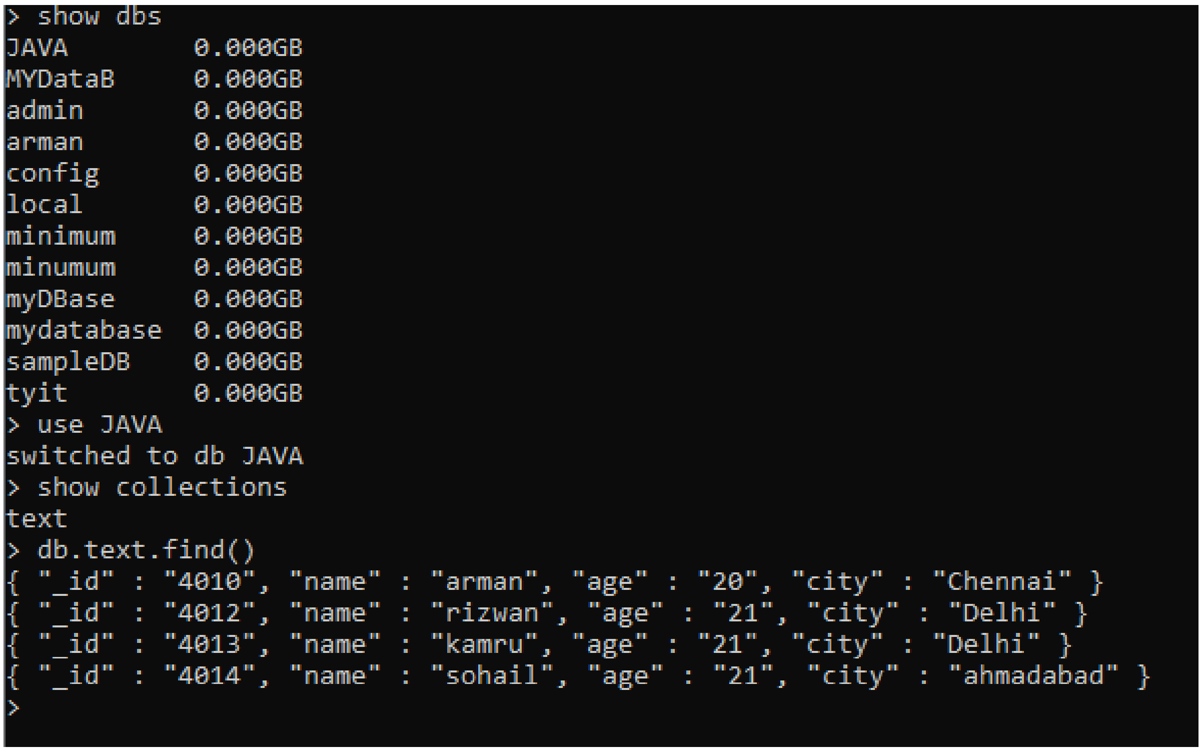
Output :

Before delete :



```
> show dbs
JAVA         0.000GB
MYDataB      0.000GB
admin        0.000GB
arman        0.000GB
config       0.000GB
local        0.000GB
minimum      0.000GB
minumum      0.000GB
myDBase      0.000GB
mydatabase   0.000GB
sampleDB     0.000GB
tyit         0.000GB
> use JAVA
switched to db JAVA
> show collections
text
> db.text.find()
{ "_id" : "4010", "name" : "arman", "age" : "20", "city" : "Chennai" }
{ "_id" : "4011", "name" : "irfan", "age" : "20", "city" : "mumbai" }
{ "_id" : "4012", "name" : "rizwan", "age" : "21", "city" : "Delhi" }
{ "_id" : "4013", "name" : "kamru", "age" : "21", "city" : "Delhi" }
{ "_id" : "4014", "name" : "sohail", "age" : "21", "city" : "ahmadabad" }
```

After Delete :



```
> show dbs
JAVA         0.000GB
MYDataB      0.000GB
admin        0.000GB
arman        0.000GB
config       0.000GB
local        0.000GB
minimum      0.000GB
minumum      0.000GB
myDBase      0.000GB
mydatabase   0.000GB
sampleDB     0.000GB
tyit         0.000GB
> use JAVA
switched to db JAVA
> show collections
text
> db.text.find()
{ "_id" : "4010", "name" : "arman", "age" : "20", "city" : "Chennai" }
{ "_id" : "4012", "name" : "rizwan", "age" : "21", "city" : "Delhi" }
{ "_id" : "4013", "name" : "kamru", "age" : "21", "city" : "Delhi" }
{ "_id" : "4014", "name" : "sohail", "age" : "21", "city" : "ahmadabad" }
>
```

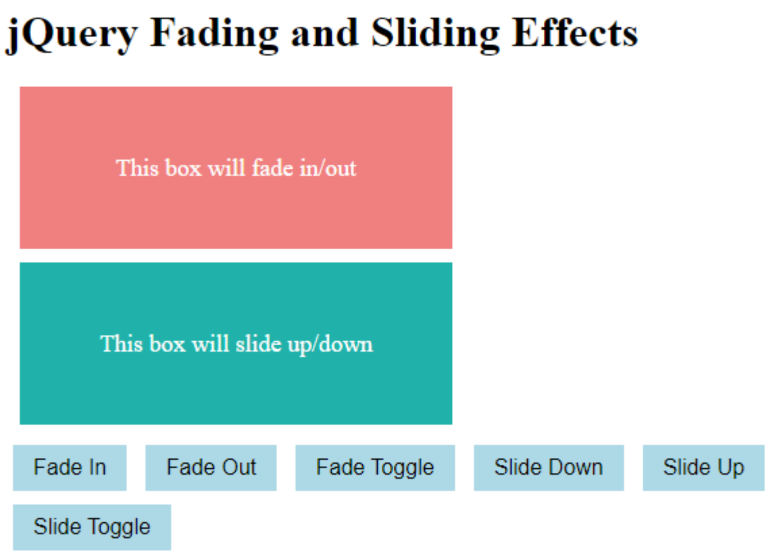
PRACTICAL NO:06

Aim : Write a program to demonstrate jQuery fading effects, jQuery Sliding effects

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>jQuery Fading and Sliding Effects Demo</title>
<!-- jQuery CDN -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<style>
.box {
width: 300px; height: 100px;
background-color: lightcoral; margin: 10px;
padding: 10px; text-align: center; line-height: 100px; font-size: 18px; color: white;
}
.effectButton { margin: 5px; padding: 8px 15px; cursor: pointer;
background-color: lightblue; border: none;
font-size: 16px;
}
</style>
</head>
<body>

<h1>jQuery Fading and Sliding Effects</h1>

<!-- Box element to demonstrate fading and sliding effects -->
<div id="fadeBox" class="box">This box will fade in/out</div>
<div id="slideBox" class="box" style="background-color: lightseagreen;">This box will slide
up/down</div>

<!-- Buttons to trigger fading effects -->
<button class="effectButton" id="fadeInButton">Fade In</button>
<button class="effectButton" id="fadeOutButton">Fade Out</button>

<button class="effectButton" id="fadeToggleButton">Fade Toggle</button>

<!-- Buttons to trigger sliding effects -->
<button class="effectButton" id="slideDownButton">Slide Down</button>
<button class="effectButton" id="slideUpButton">Slide Up</button>
<button class="effectButton" id="slideToggleButton">Slide Toggle</button>

<script>
$(document).ready(function(){
```

```
// Fading Effects
$("#fadeInButton").click(function(){
$("#fadeBox").fadeIn();
});

$("#fadeOutButton").click(function(){
$("#fadeBox").fadeOut();
});

$("#fadeToggleButton").click(function(){
$("#fadeBox").fadeToggle();
});

// Sliding Effects
$("#slideDownButton").click(function(){
$("#slideBox").slideDown();
});

$("#slideUpButton").click(function(){
$("#slideBox").slideUp();
});

$("#slideToggleButton").click(function(){
$("#slideBox").slideToggle();
});
});
</script>

</body>
</html>
```

Output:


jQuery Fading and Sliding Effects

This box will fade in/out

This box will slide up/down

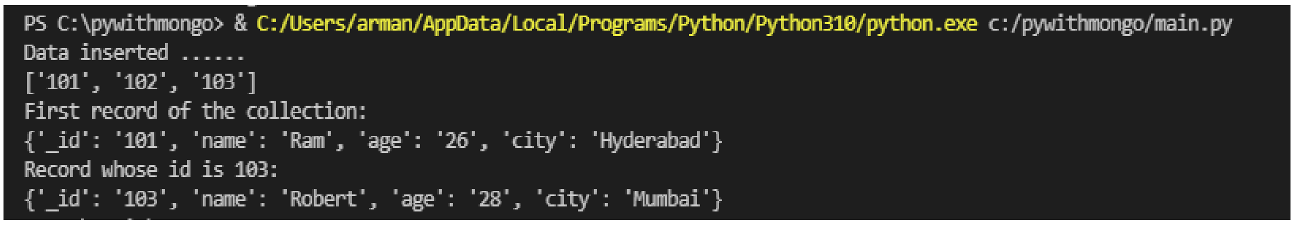
PRACTICAL NO:07

PYTHON AND MONGODB

a) Connecting Python with MongoDB and inserting, retrieving, updating and deleting.

1) Inserting & Retrieving Source code:
from pymongo import MongoClient
#Creating a pymongo client client = MongoClient('localhost', 27017)
#Getting the database #instance
db = client['mydatabase'] #Creating a collection coll = db['example']
#Inserting document into a collection data = [{"_id": "101", "name": "Ram", "age": "26", "city":
"Hyderabad"}, {"_id": "102", "name": "Rahim", "age": "27", "city":
"Bangalore"}, {"_id": "103", "name": "Robert", "age": "28", "city":
"Mumbai"}]
res = coll.insert_many(data) print("Data inserted ")
print(res.inserted_ids)
#Retrieving the first record using the find_one() method print("First record of the collection:
") print(coll.find_one()) #Retrieving a record with is 103 using #thefind_one()
method print("Record whose id is 103: ") print(coll.find_one({"_id": "103"}))

OUTPUT :

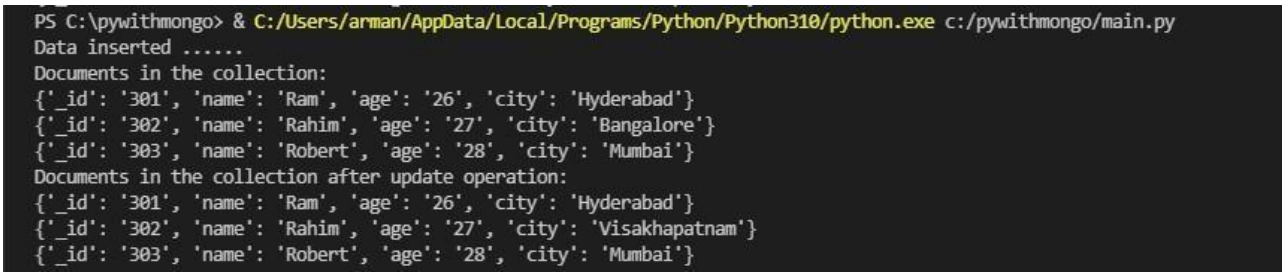


```
PS C:\pywithmongo> & C:/Users/arman/AppData/Local/Programs/Python/Python310/python.exe c:/pywithmongo/main.py
Data inserted ......
['101', '102', '103']
First record of the collection:
{'_id': '101', 'name': 'Ram', 'age': '26', 'city': 'Hyderabad'}
Record whose id is 103:
{'_id': '103', 'name': 'Robert', 'age': '28', 'city': 'Mumbai'}
```

2. Update a.Update One Source code:
from pymongo import MongoClient #Creating a pymongo client
client = MongoClient('localhost', 27017) #Getting the database instance
db = client['myDBase'] #Creating a collection coll = db['MYExample2']
#Inserting document into a collection data = [
{"_id": "301", "name": "Ram", "age": "26", "city": "Hyderabad"},
{"_id": "302", "name": "Rahim", "age": "27", "city":
"Bangalore"},
{"_id": "303", "name": "Robert", "age": "28", "city": "Mumbai"}
]
res = coll.insert_many(data) print("Data inserted ")
#Retrieving all the records using the find() method print("Documents in the collection: ")

for doc1 in coll.find(): print(doc1)
coll.update_one({"_id":"302"},{"$set":{"city":"Visakhapatnam"}
})
#Retrieving all the records using the find() method print("Documents in the collection after update operation: ") for doc2 in coll.find():
print(doc2)

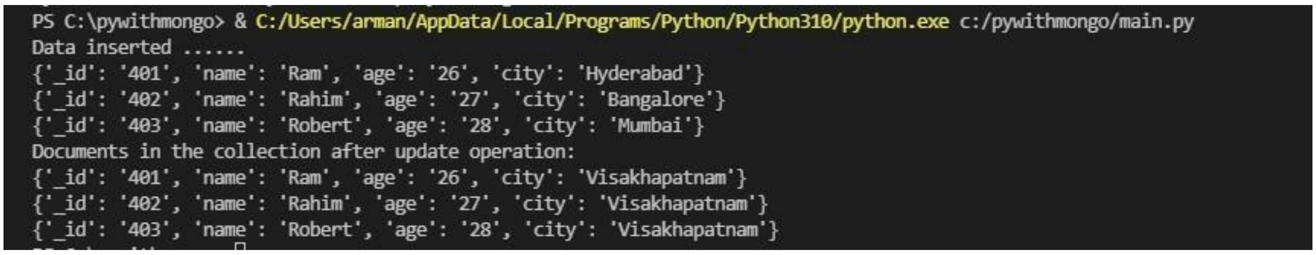
OUTPUT :


```
PS C:\pywithmongo> & C:/Users/arman/AppData/Local/Programs/Python/Python310/python.exe c:/pywithmongo/main.py
Data inserted ......
Documents in the collection:
{'_id': '301', 'name': 'Ram', 'age': '26', 'city': 'Hyderabad'}
{'_id': '302', 'name': 'Rahim', 'age': '27', 'city': 'Bangalore'}
{'_id': '303', 'name': 'Robert', 'age': '28', 'city': 'Mumbai'}
Documents in the collection after update operation:
{'_id': '301', 'name': 'Ram', 'age': '26', 'city': 'Hyderabad'}
{'_id': '302', 'name': 'Rahim', 'age': '27', 'city': 'Visakhapatnam'}
{'_id': '303', 'name': 'Robert', 'age': '28', 'city': 'Mumbai'}
```

b) Update Many Source code:

from pymongo import MongoClient #Creating a pymongo client
client = MongoClient('localhost', 27017) #Getting the database instance
db = client['MYDataB'] #Creating a collection coll = db['MYexample5']
#Inserting document into a collection data = [
{"_id": "401", "name": "Ram", "age": "26", "city": "Hyderabad"},
{"_id": "402", "name": "Rahim", "age": "27", "city": "Bangalore"},
{"_id": "403", "name": "Robert", "age": "28", "city": "Mumbai"}
]
res = coll.insert_many(data) print("Data inserted ")
#Retrieving all the records using the find() method print("Documents in the collection: ")
for doc1 in coll.find(): print(doc1)
coll.update_many({},{"$set":{"city":"Visakhapatnam"}}) #Retrieving all the records using the find() method print("Documents in the collection after update operation: ") for doc2 in coll.find():
print(doc2)

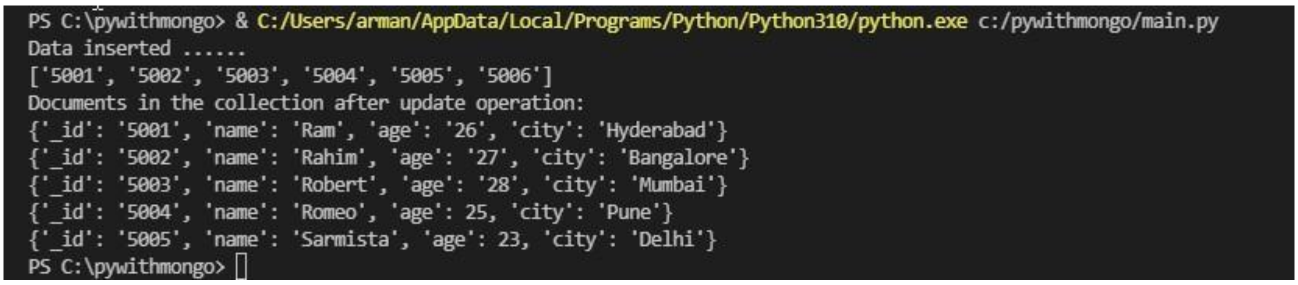
OUTPUT :


```
PS C:\pywithmongo> & C:/Users/arman/AppData/Local/Programs/Python/Python310/python.exe c:/pywithmongo/main.py
Data inserted ......
{'_id': '401', 'name': 'Ram', 'age': '26', 'city': 'Hyderabad'}
{'_id': '402', 'name': 'Rahim', 'age': '27', 'city': 'Bangalore'}
{'_id': '403', 'name': 'Robert', 'age': '28', 'city': 'Mumbai'}
Documents in the collection after update operation:
{'_id': '401', 'name': 'Ram', 'age': '26', 'city': 'Visakhapatnam'}
{'_id': '402', 'name': 'Rahim', 'age': '27', 'city': 'Visakhapatnam'}
{'_id': '403', 'name': 'Robert', 'age': '28', 'city': 'Visakhapatnam'}
```

3. Delete a.Delete One Source code:

```
from pymongo import MongoClient #Creating a pymongo client
client = MongoClient('localhost', 27017)
#Getting the database instance db = client['mydatabase'] #Creating a collection
coll = db['Myexample']
#Inserting document into a collection data = [
{"_id": "5001", "name": "Ram", "age": "26", "city": "Hyderabad"},
{"_id": "5002", "name": "Rahim", "age": "27", "city": "Bangalore"},
{"_id": "5003", "name": "Robert", "age": "28", "city": "Mumbai"},
{"_id": "5004", "name": "Romeo", "age": 25, "city": "Pune"},
{"_id": "5005", "name": "Sarmista", "age": 23, "city": "Delhi"},
{"_id": "5006", "name": "Rasajna", "age": 26, "city": "Chennai"}
]
res = coll.insert_many(data) print("Data inserted     ")
print(res.inserted_ids) #Deleting one document coll.delete_one({"_id" : "5006"})
#Retrieving all the records using the find() method print("Documents in the collection after
update operation: ") for doc2 in coll.find():
print(doc2)
```

OUTPUT :



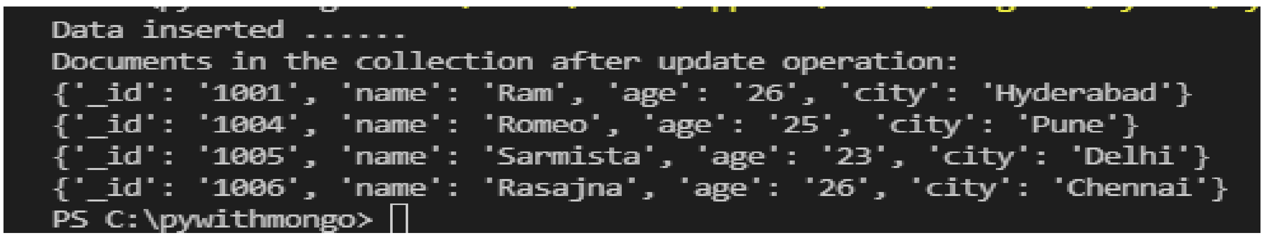
```
PS C:\pywithmongo> & C:/Users/arman/AppData/Local/Programs/Python/Python310/python.exe c:/pywithmongo/main.py
Data inserted ......
['5001', '5002', '5003', '5004', '5005', '5006']
Documents in the collection after update operation:
{'_id': '5001', 'name': 'Ram', 'age': '26', 'city': 'Hyderabad'}
{'_id': '5002', 'name': 'Rahim', 'age': '27', 'city': 'Bangalore'}
{'_id': '5003', 'name': 'Robert', 'age': '28', 'city': 'Mumbai'}
{'_id': '5004', 'name': 'Romeo', 'age': 25, 'city': 'Pune'}
{'_id': '5005', 'name': 'Sarmista', 'age': 23, 'city': 'Delhi'}
PS C:\pywithmongo> 
```

a. Delete Many Source code:

```
from pymongo import MongoClient
#Creating a pymongo client client = MongoClient('localhost',
27017) #Getting the database instance db = client['sampleDB'] #Creating a collection coll =
db['example 4']
#Inserting document into data = [
{"_id": "1001", "name": "Ram", "age": "26", "city": "Hyderabad"},
{"_id": "1002", "name": "Rahim", "age": "27", "city": "Bangalore"},
{"_id": "1003", "name": "Robert", "age": "28", "city": "Mumbai"},
{"_id": "1004", "name": "Romeo", "age": "25", "city":
"Pune"}, {"_id": "1005", "name": "Sarmista", "age": "23", "city":
"Delhi"},
{"_id": "1006", "name": "Rasajna", "age": "26", "city": "Chennai"}
]
res = coll.insert_many(data) print("Data inserted     ")
#Deleting multiple documents coll.delete_many({"age":{"$gt":"26"}})
```

#Retrieving all the records using the find() method print("Documents in the collection after update operation: ")
for doc2 in coll.find(): print(doc2)

OUTPUT :



```
Data inserted ......
Documents in the collection after update operation:
{'_id': '1001', 'name': 'Ram', 'age': '26', 'city': 'Hyderabad'}
{'_id': '1004', 'name': 'Romeo', 'age': '25', 'city': 'Pune'}
{'_id': '1005', 'name': 'Sarmista', 'age': '23', 'city': 'Delhi'}
{'_id': '1006', 'name': 'Rasajna', 'age': '26', 'city': 'Chennai'}
PS C:\pywithmongo>
```

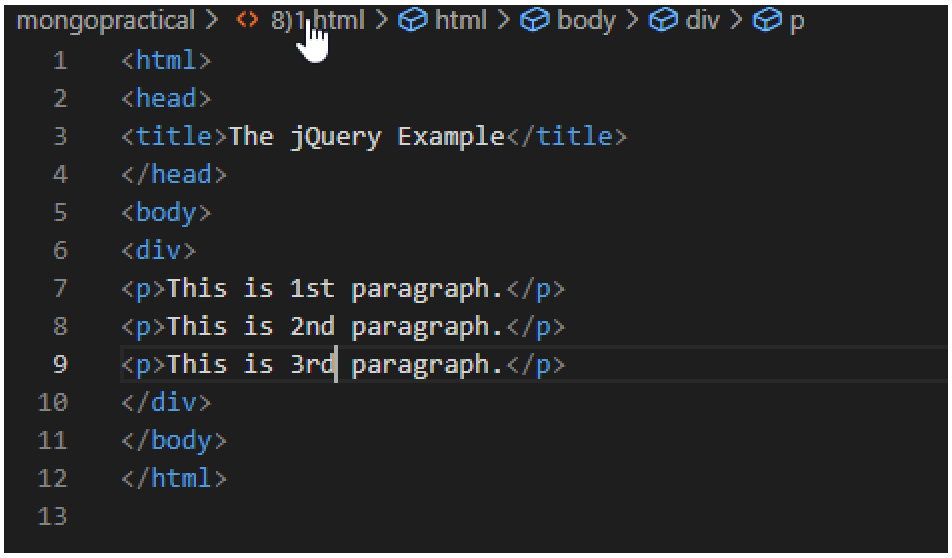
PRACTICAL NO:08

PROGRAMS ON BASIC JQUERY

a) Query Basic, jQuery Events.

Jquery basic :

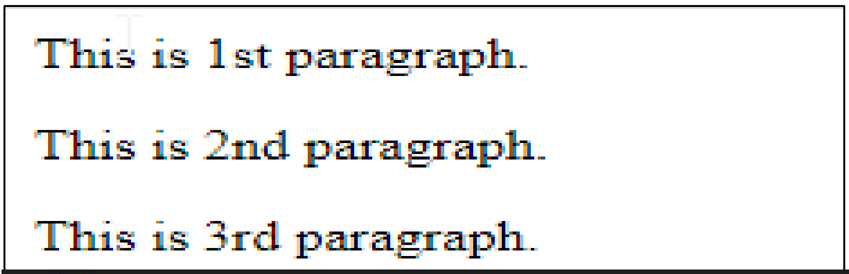
CODE :



mongopractical > 8)1.html > html > body > div > p

```html
<html>
<head>
<title>The jQuery Example</title>
</head>
<body>
<div>
<p>This is 1st paragraph.</p>
<p>This is 2nd paragraph.</p>
<p>This is 3rd paragraph.</p>
</div>
</body>
</html>
```

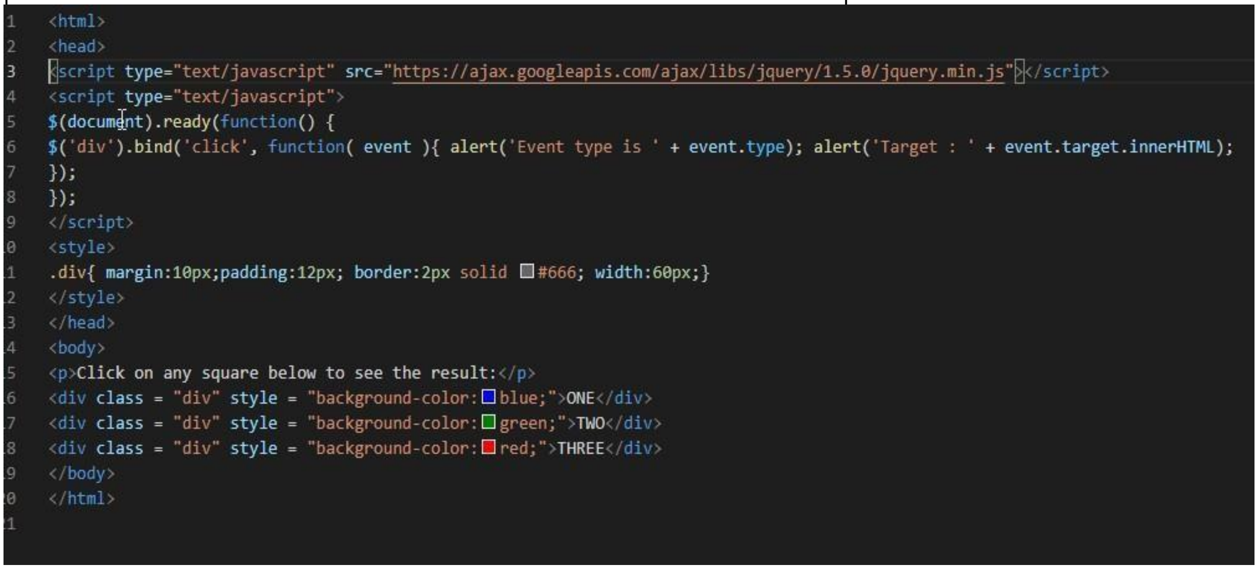
OUTPUT :



This is 1st paragraph.

This is 2nd paragraph.

This is 3rd paragraph.

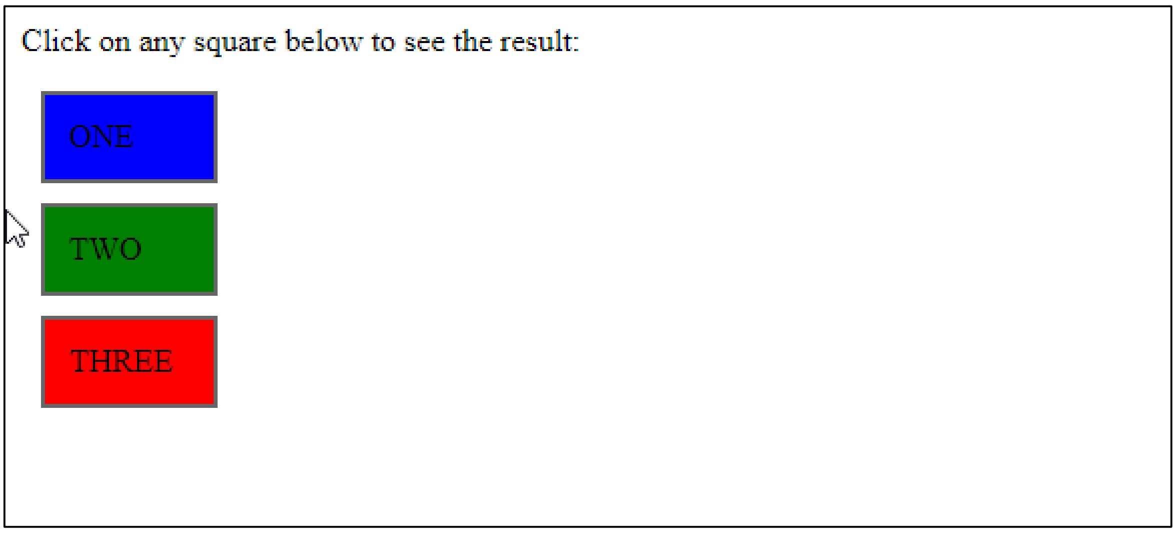


```html
<html>
<head>
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.0/jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
$('div').bind('click', function( event ){ alert('Event type is ' + event.type); alert('Target : ' + event.target.innerHTML);
});
});
</script>
<style>
.div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
</style>
</head>
<body>
<p>Click on any square below to see the result:</p>
<div class = "div" style = "background-color:blue;">ONE</div>
<div class = "div" style = "background-color:green;">TWO</div>
<div class = "div" style = "background-color:red;">THREE</div>
</body>
</html>
```

1) JQuery Events Click Event :

CODE :

OUTPUT:

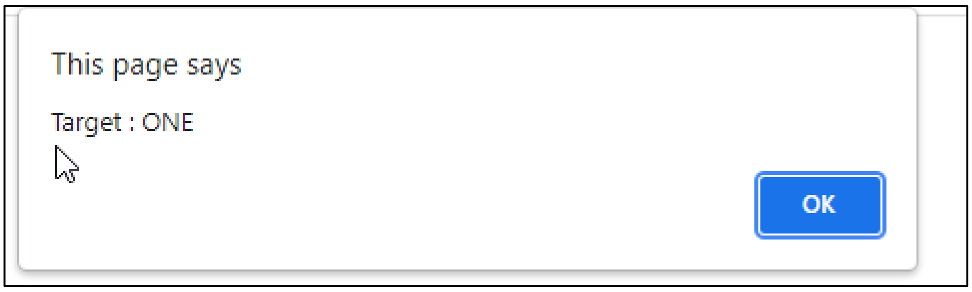


Click on any square below to see the result:

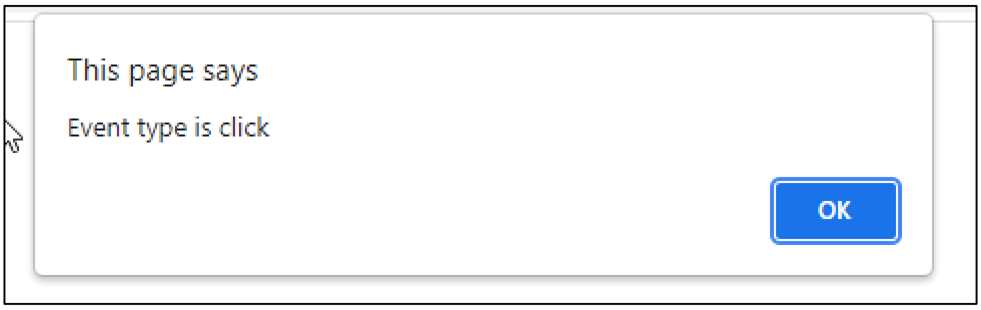
ONE

TWO

THREE


This page says

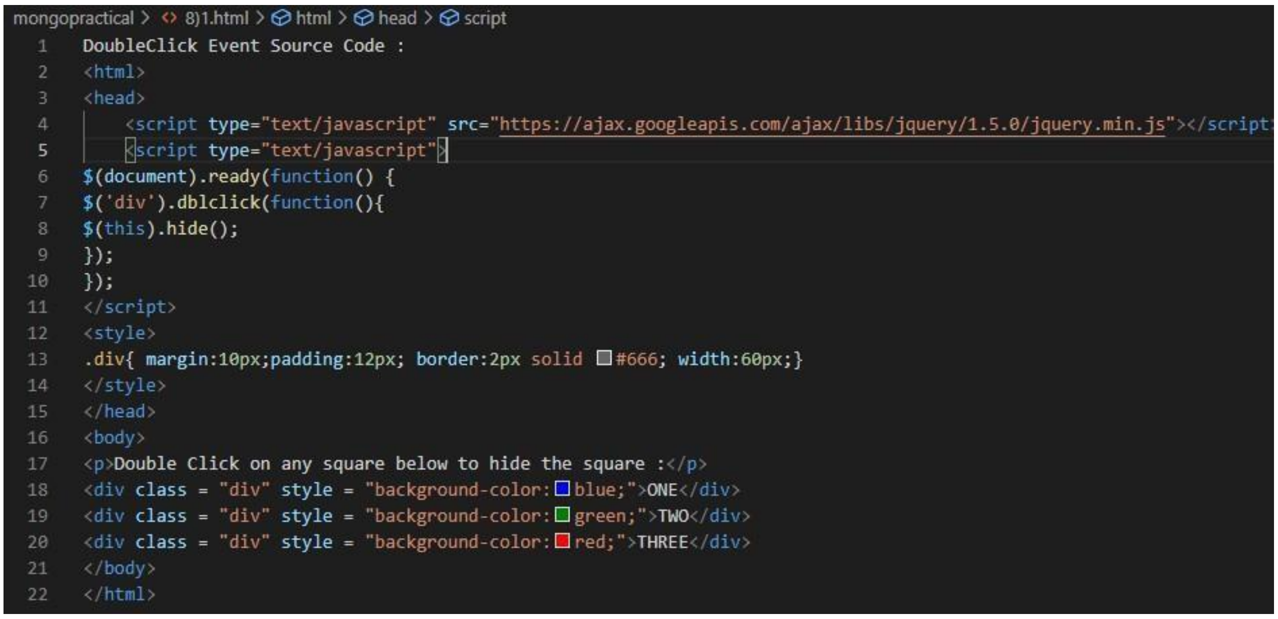
Target : ONE


This page says

Event type is click

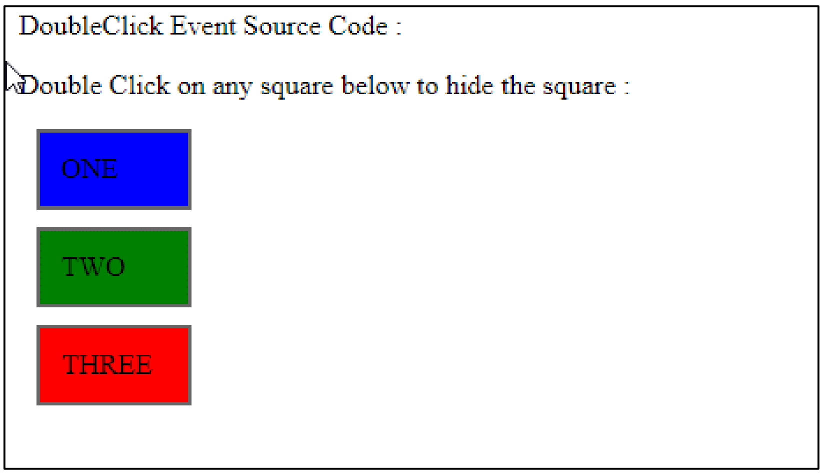
2) DoubleClick Event

Source Code :



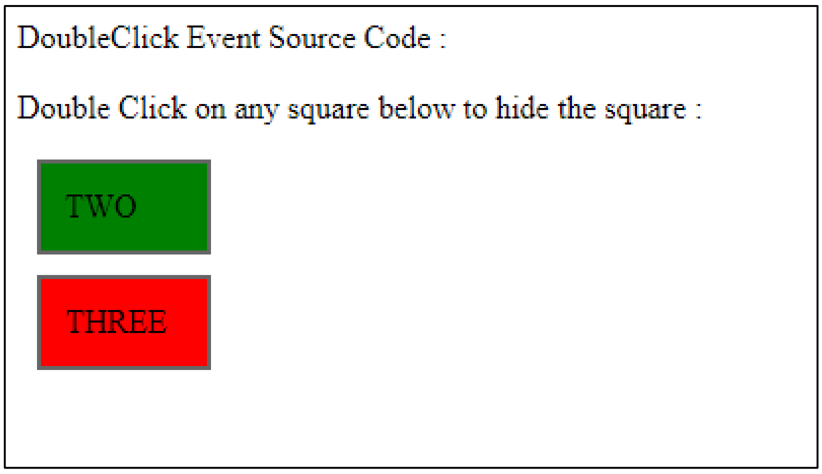
```
DoubleClick Event Source Code :
<html>
<head>
    <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.0/jquery.min.js"></script
    <script type="text/javascript">
$(document).ready(function() {
$('div').dblclick(function(){
$(this).hide();
});
});
</script>
<style>
.div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
</style>
</head>
<body>
<p>Double Click on any square below to hide the square :</p>
<div class = "div" style = "background-color:blue;">ONE</div>
<div class = "div" style = "background-color:green;">TWO</div>
<div class = "div" style = "background-color:red;">THREE</div>
</body>
</html>
```

OUTPUT :



DoubleClick Event Source Code :

Double Click on any square below to hide the square :

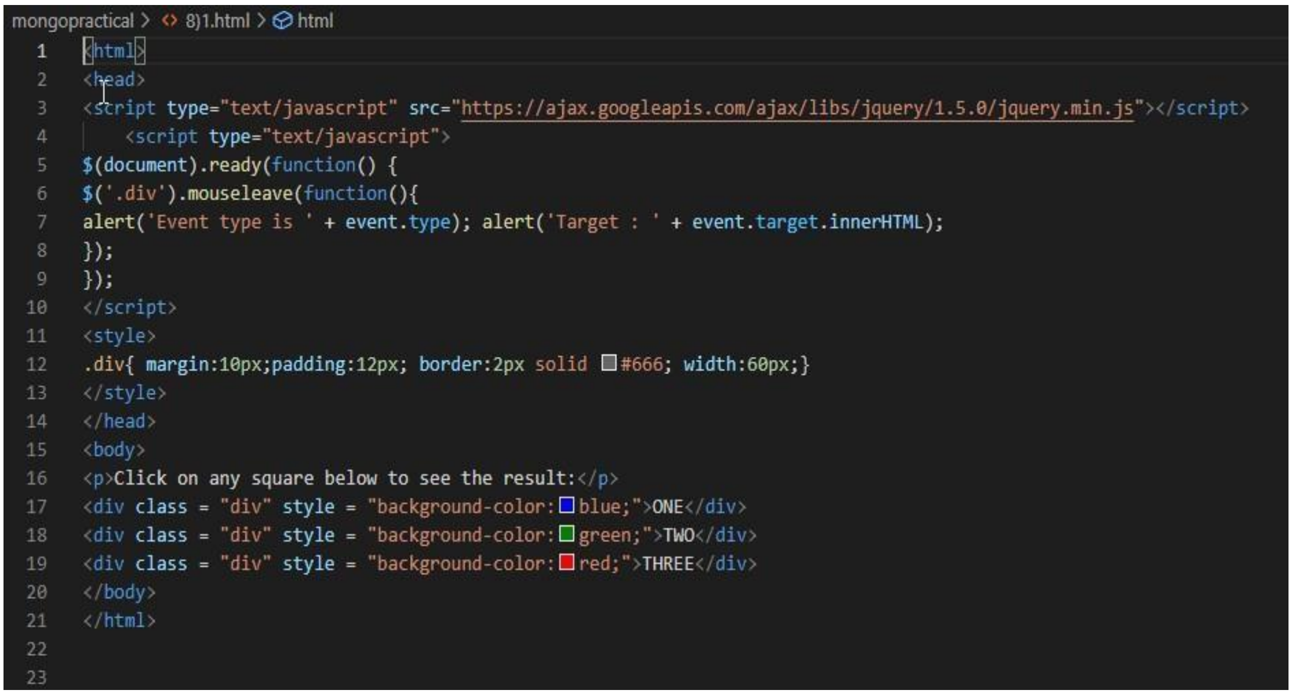


DoubleClick Event Source Code :

Double Click on any square below to hide the square :

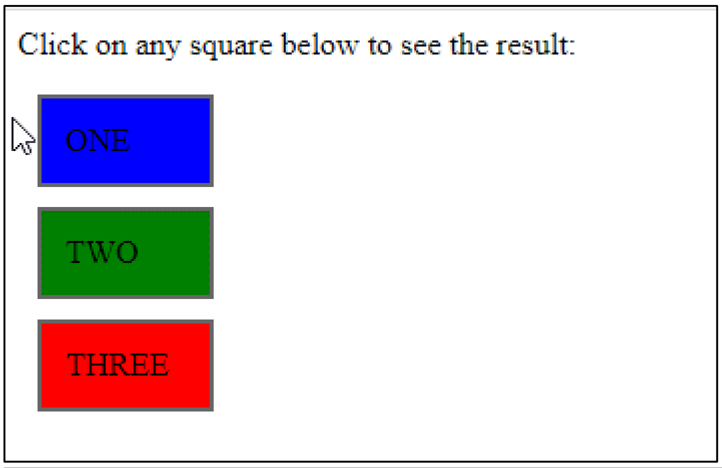
3) Mouseleave

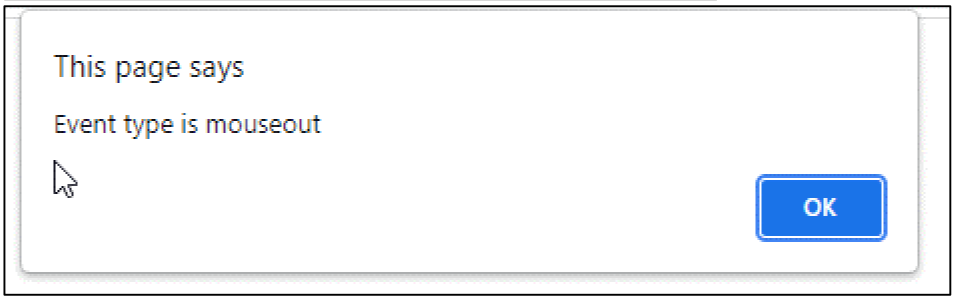
Source Code :


mongopractical > 8)1.html > html

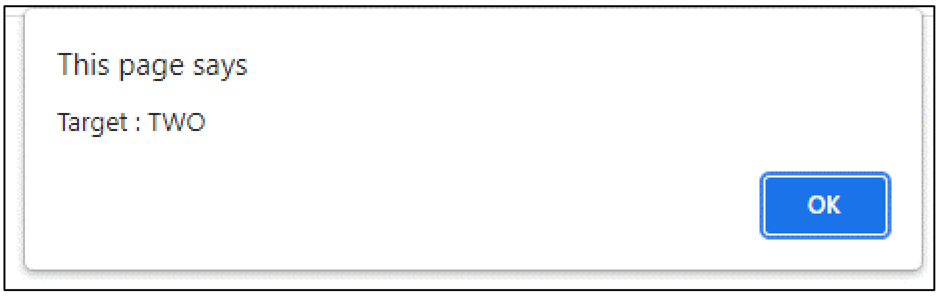
```
<html>
<head>
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.0/jquery.min.js"></script>
    <script type="text/javascript">
$(document).ready(function() {
$('.div').mouseleave(function(){
alert('Event type is ' + event.type); alert('Target : ' + event.target.innerHTML);
});
});
</script>
<style>
.div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
</style>
</head>
<body>
<p>Click on any square below to see the result:</p>
<div class = "div" style = "background-color:blue;">ONE</div>
<div class = "div" style = "background-color:green;">TWO</div>
<div class = "div" style = "background-color:red;">THREE</div>
</body>
</html>
```

OUTPUT :


Click on any square below to see the result:


This page says

Event type is mouseout

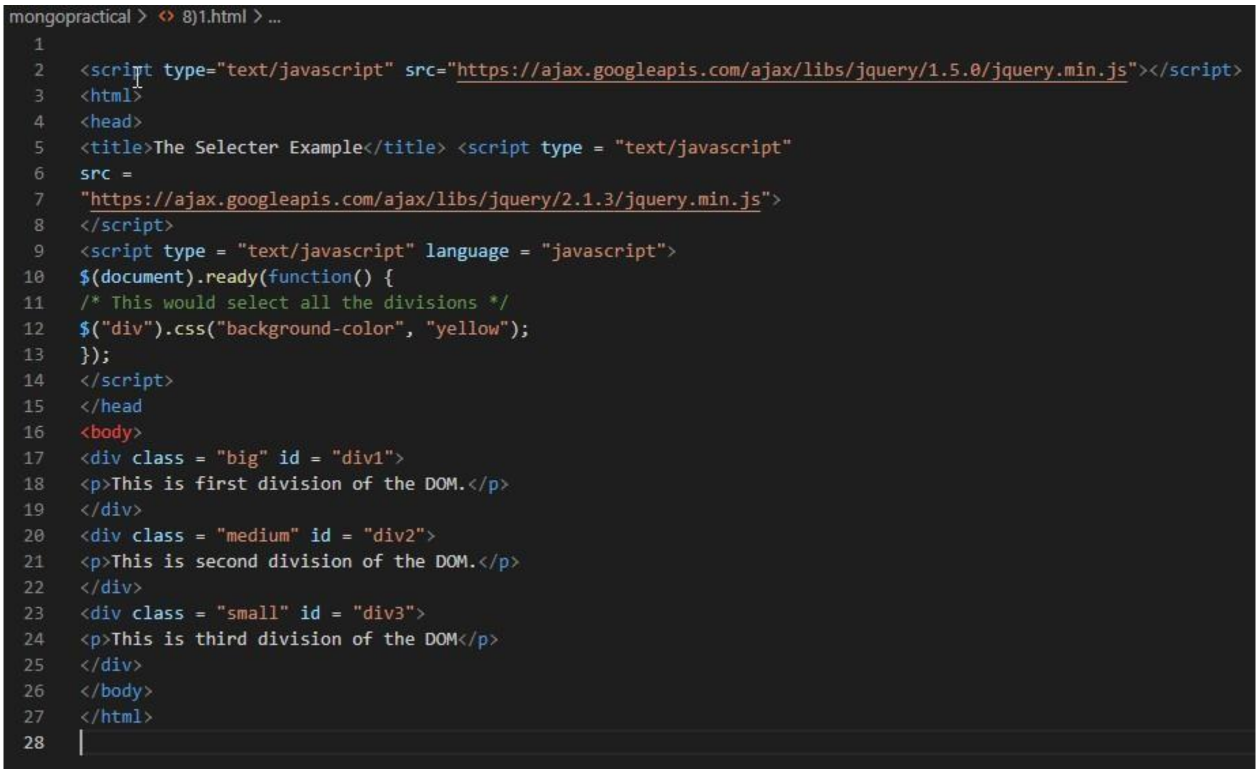

This page says

Target : TWO

b) JQuery Selectors, JQuery Hide and Show Effects JQuery Selectors –

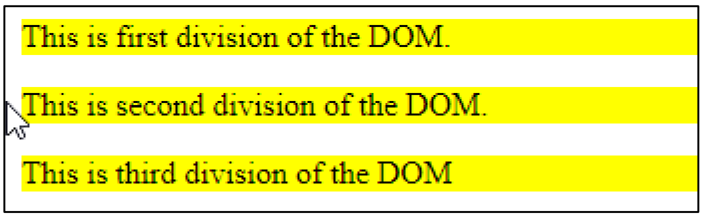
Name Selector

CODE :

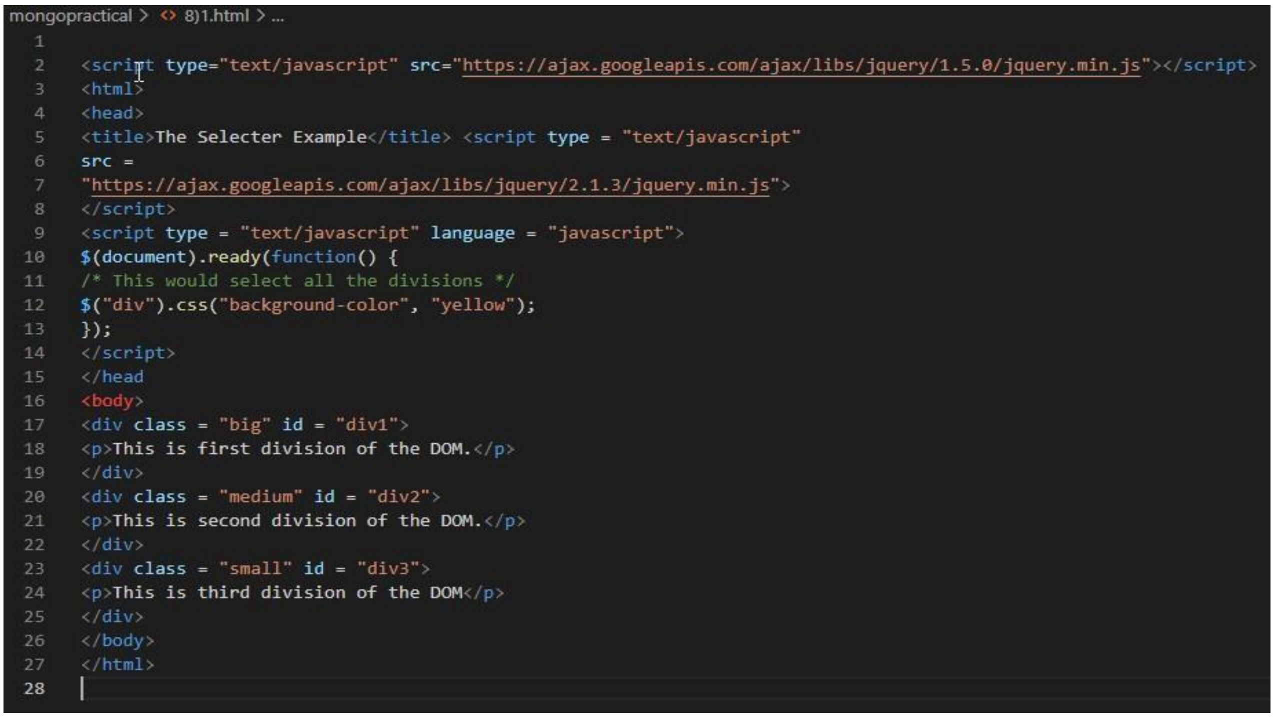


```

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.0/jquery.min.js"></script>
<html>
<head>
<title>The Selecter Example</title> <script type = "text/javascript"
src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
<script type = "text/javascript" language = "javascript">
$(document).ready(function() {
/* This would select all the divisions */
$("div").css("background-color", "yellow");
});
</script>
</head
<body>
<div class = "big" id = "div1">
<p>This is first division of the DOM.</p>
</div>
<div class = "medium" id = "div2">
<p>This is second division of the DOM.</p>
</div>
<div class = "small" id = "div3">
<p>This is third division of the DOM</p>
</div>
</body>
</html>

```

OUTPUT :


```
This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM
```

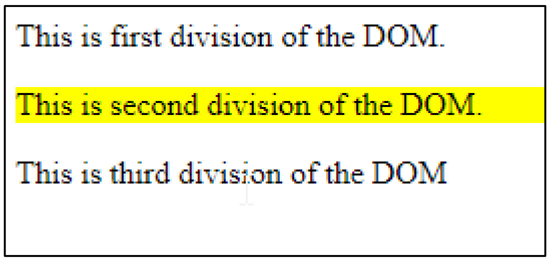
ID Selector Source Code :



mongopractical > 8)1.html > ...

```html

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1.5.0/jquery.min.js"></script>
<html>
<head>
<title>The Selecter Example</title> <script type = "text/javascript"
src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
<script type = "text/javascript" language = "javascript">
$(document).ready(function() {
/* This would select all the divisions */
$("div").css("background-color", "yellow");
});
</script>
</head
<body>
<div class = "big" id = "div1">
<p>This is first division of the DOM.</p>
</div>
<div class = "medium" id = "div2">
<p>This is second division of the DOM.</p>
</div>
<div class = "small" id = "div3">
<p>This is third division of the DOM</p>
</div>
</body>
</html>

```

OUTPUT :

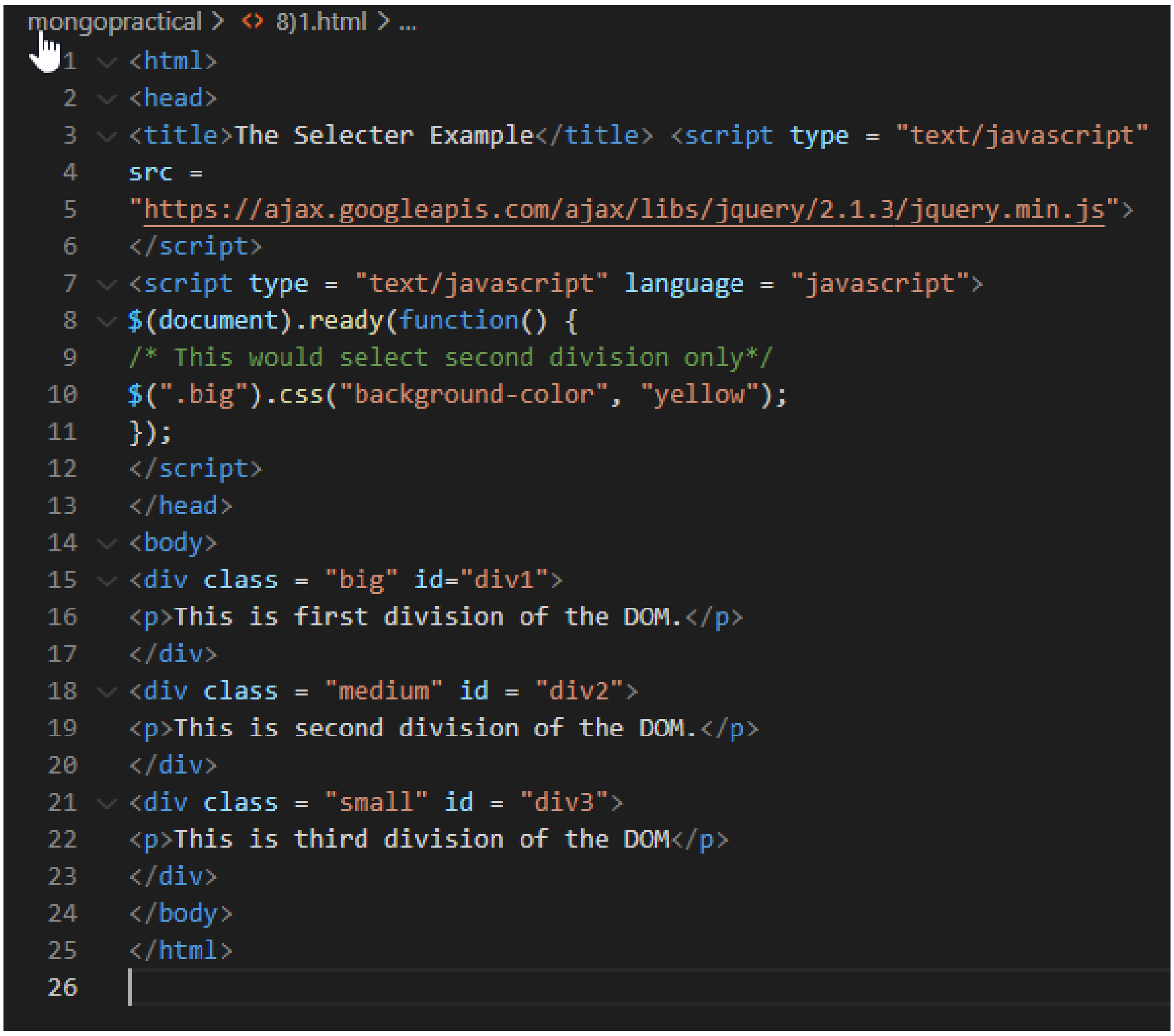


This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

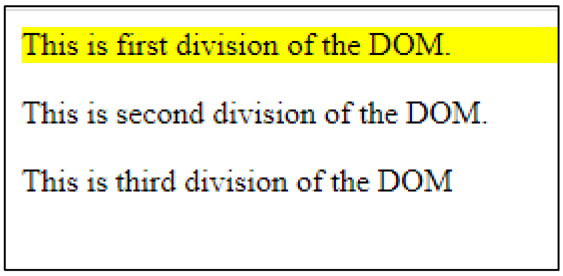
Class Selector Source Code :



mongopractical > <> 8)1.html > ...

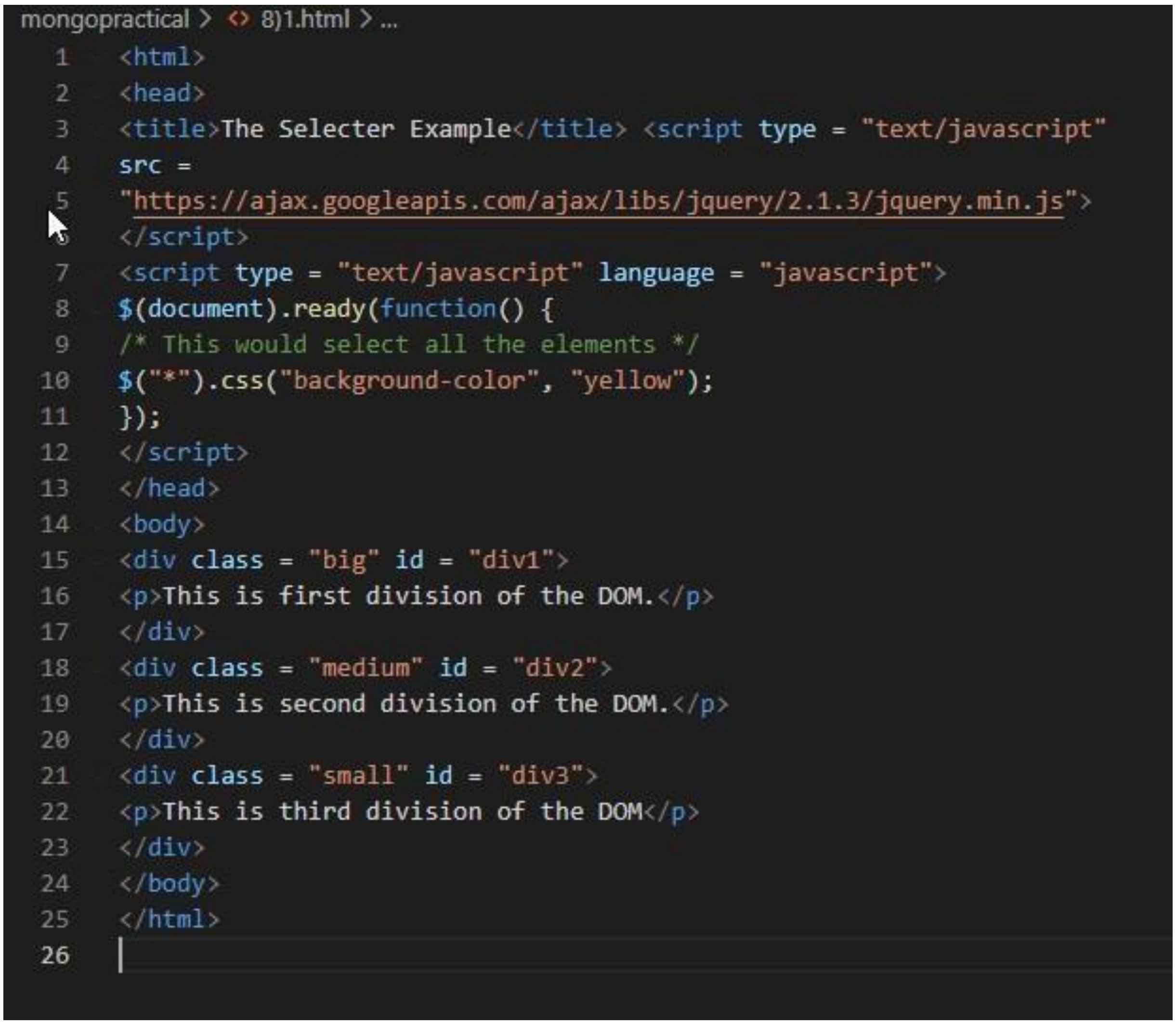
```html
<html>
<head>
<title>The Selecter Example</title> <script type = "text/javascript"
src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
<script type = "text/javascript" language = "javascript">
$(document).ready(function() {
/* This would select second division only*/
$(".big").css("background-color", "yellow");
});
</script>
</head>
<body>
<div class = "big" id="div1">
<p>This is first division of the DOM.</p>
</div>
<div class = "medium" id = "div2">
<p>This is second division of the DOM.</p>
</div>
<div class = "small" id = "div3">
<p>This is third division of the DOM</p>
</div>
</body>
</html>
```

OUTPUT :



```
This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM
```

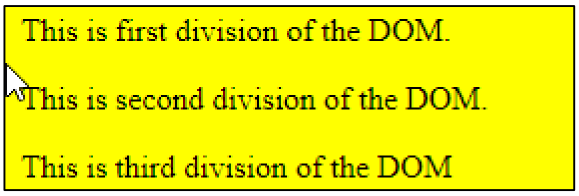
Universal selector Source Code :



mongopractical > <> 8)1.html > ...

```
<html>
<head>
<title>The Selecter Example</title> <script type = "text/javascript"
src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
<script type = "text/javascript" language = "javascript">
$(document).ready(function() {
/* This would select all the elements */
$("*").css("background-color", "yellow");
});
</script>
</head>
<body>
<div class = "big" id = "div1">
<p>This is first division of the DOM.</p>
</div>
<div class = "medium" id = "div2">
<p>This is second division of the DOM.</p>
</div>
<div class = "small" id = "div3">
<p>This is third division of the DOM</p>
</div>
</body>
</html>
```

OUTPUT :

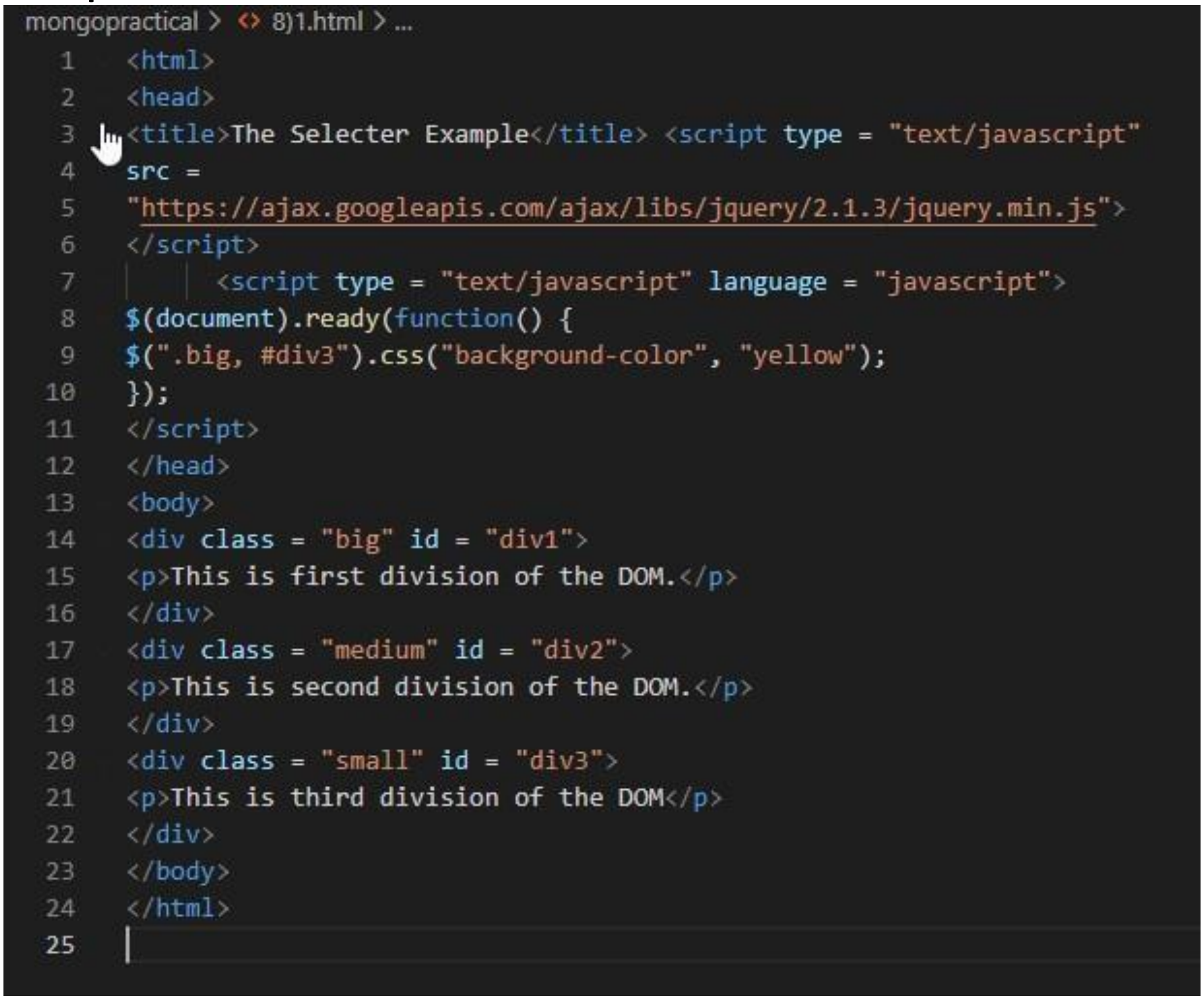


This is first division of the DOM.

This is second division of the DOM.

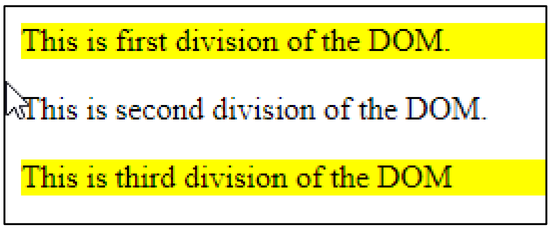
This is third division of the DOM

Multiple Selector Source Code :


mongopractical > <> 8)1.html > ...

```
<html>
<head>
<title>The Selecter Example</title> <script type = "text/javascript"
src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
        <script type = "text/javascript" language = "javascript">
$(document).ready(function() {
$(".big, #div3").css("background-color", "yellow");
});
</script>
</head>
<body>
<div class = "big" id = "div1">
<p>This is first division of the DOM.</p>
</div>
<div class = "medium" id = "div2">
<p>This is second division of the DOM.</p>
</div>
<div class = "small" id = "div3">
<p>This is third division of the DOM</p>
</div>
</body>
</html>
```

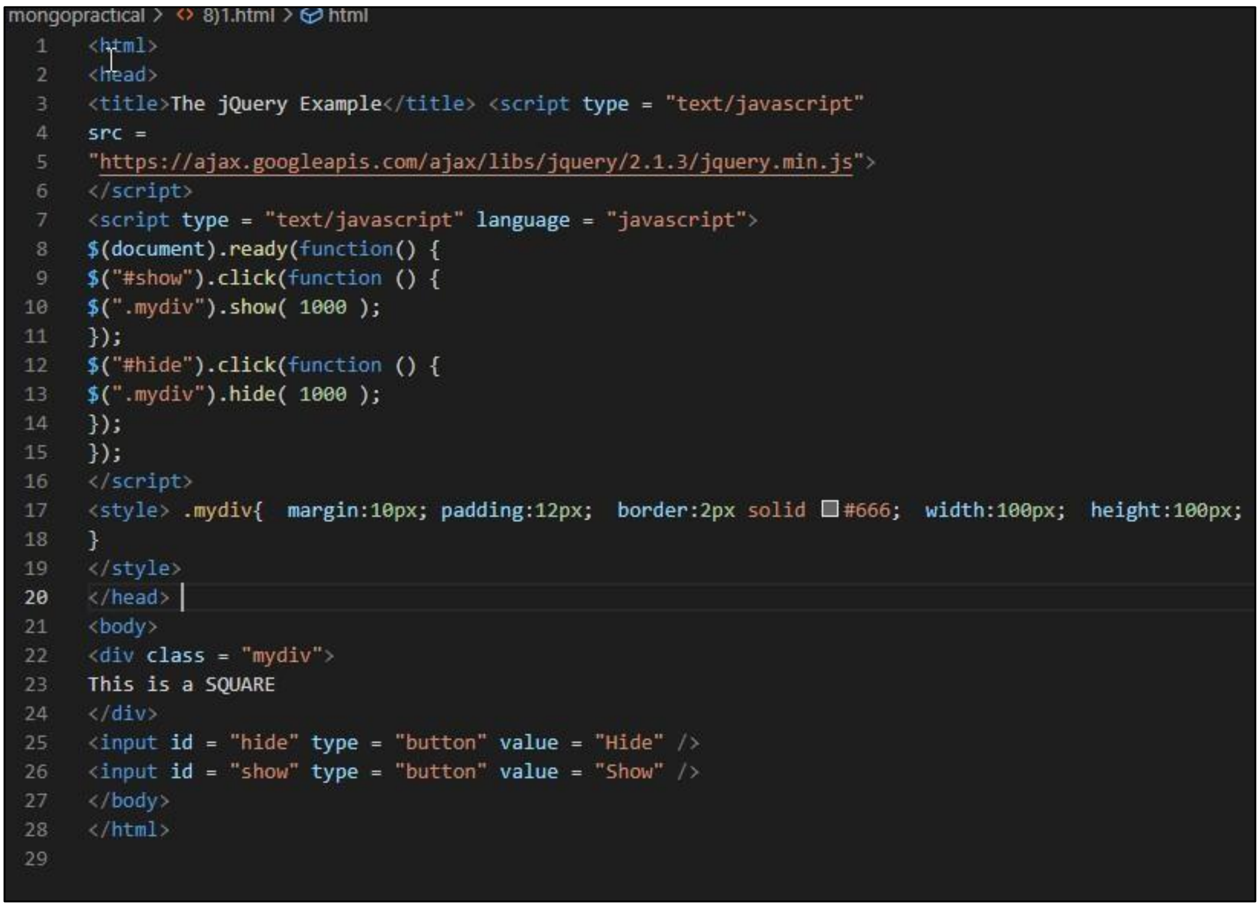
OUTPUT :


This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

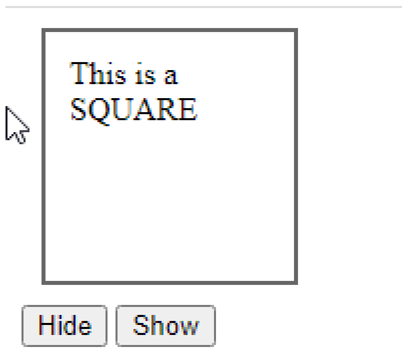
Jquery hide and show effect source : Code :

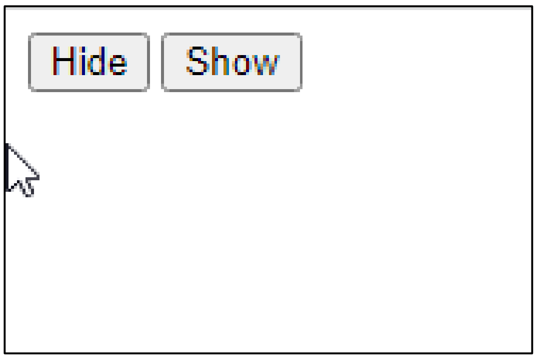


mongopractical > 8)1.html > html

```html
<html>
<head>
<title>The jQuery Example</title> <script type = "text/javascript"
src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
<script type = "text/javascript" language = "javascript">
$(document).ready(function() {
$("#show").click(function () {
$(".mydiv").show( 1000 );
});
$("#hide").click(function () {
$(".mydiv").hide( 1000 );
});
});
</script>
<style> .mydiv{  margin:10px; padding:12px;  border:2px solid #666;  width:100px;  height:100px;
}
</style>
</head>
<body>
<div class = "mydiv">
This is a SQUARE
</div>
<input id = "hide" type = "button" value = "Hide" />
<input id = "show" type = "button" value = "Show" />
</body>
</html>

```

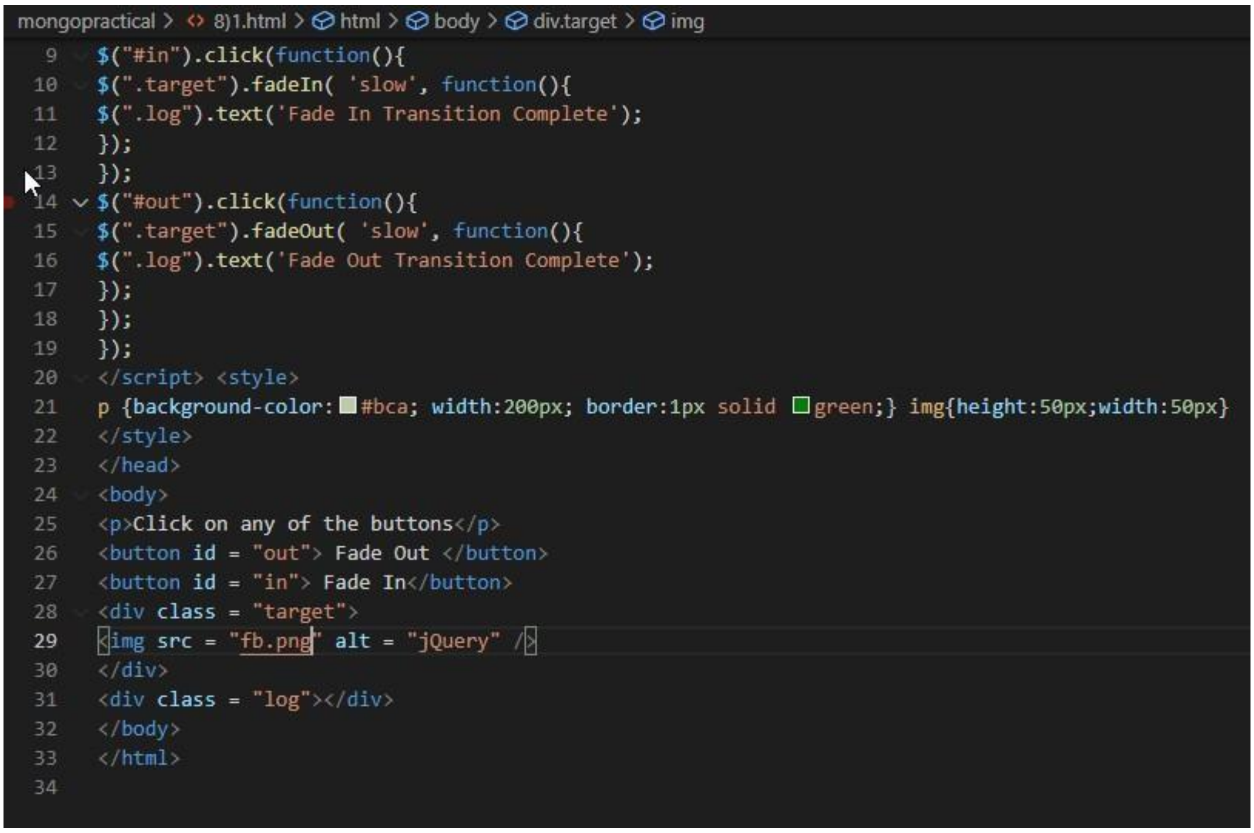
OUTPUT :





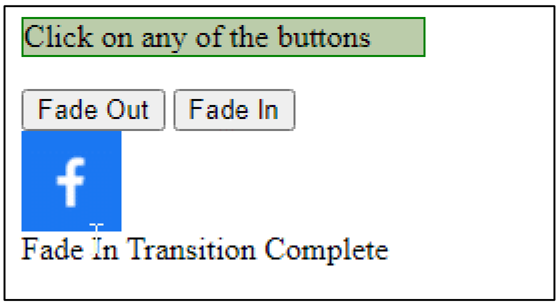
C. JQuery Fading Effect, JQuery Sliding Effect

JQuery Fading Effect
Source Code :

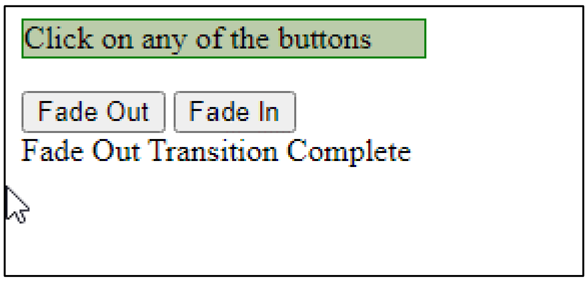


```
mongopractical > <> 8)1.html > html > body > div.target > img
$("#in").click(function(){
$(".target").fadeIn( 'slow', function(){
$(".log").text('Fade In Transition Complete');
});
});
$("#out").click(function(){
$(".target").fadeOut( 'slow', function(){
$(".log").text('Fade Out Transition Complete');
});
});
});
</script> <style>
p {background-color:#bca; width:200px; border:1px solid green;} img{height:50px;width:50px}
</style>
</head>
<body>
<p>Click on any of the buttons</p>
<button id = "out"> Fade Out </button>
<button id = "in"> Fade In</button>
<div class = "target">
<img src = "fb.png" alt = "jQuery" />
</div>
<div class = "log"></div>
</body>
</html>
```

OUTPUT :

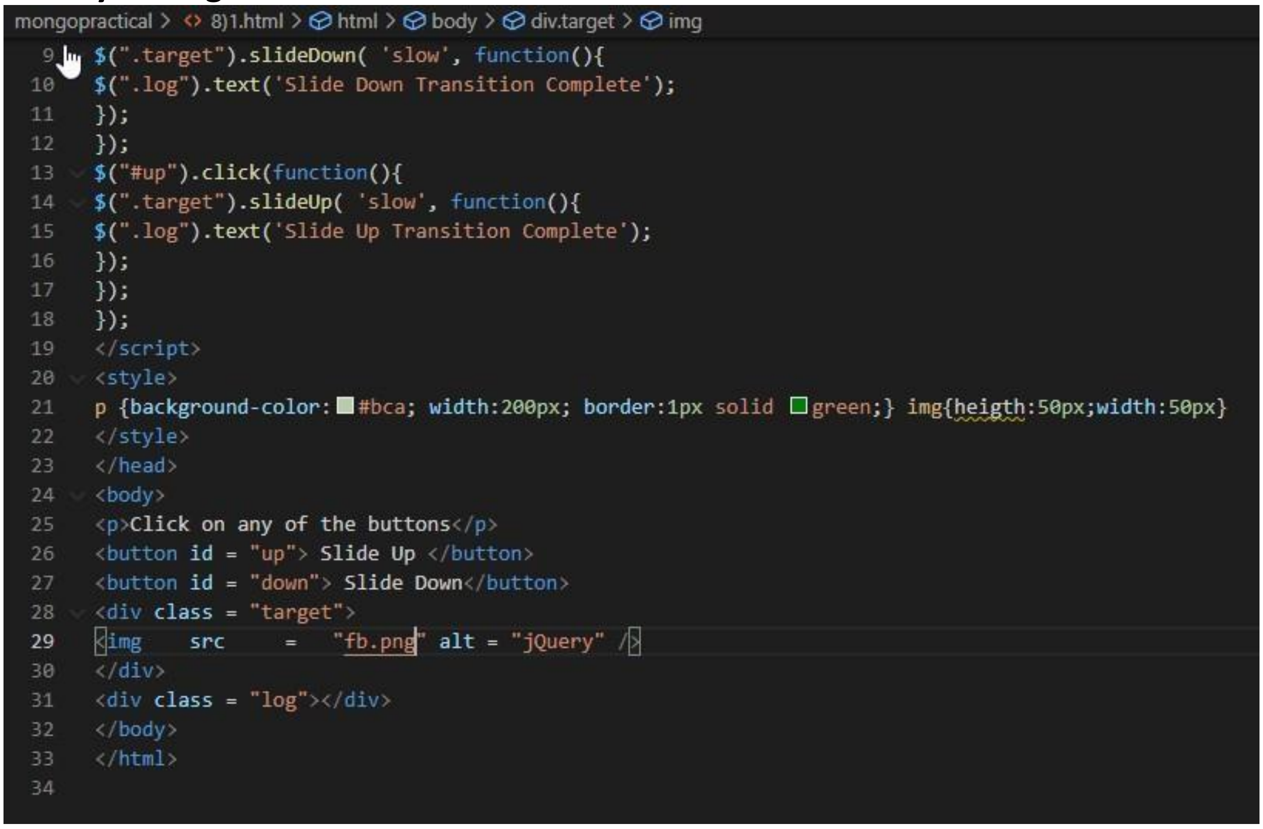


```
Click on any of the buttons
Fade Out  Fade In
Fade In Transition Complete
```



```
Click on any of the buttons
Fade Out  Fade In
Fade Out Transition Complete
```

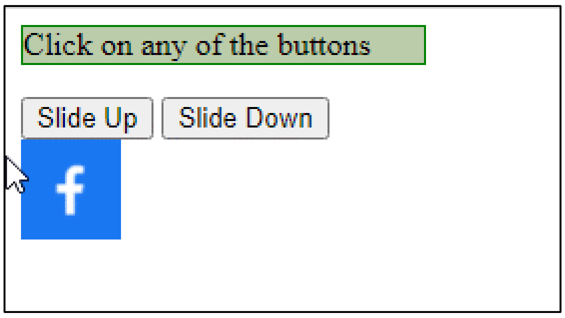
JQuery Sliding Effect Source Code :



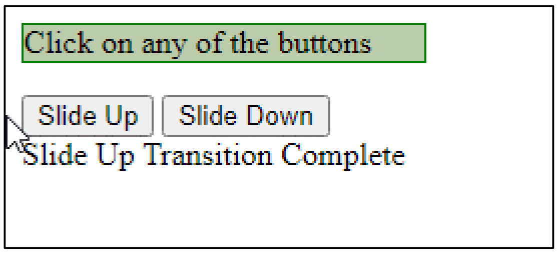
mongopractical > 8)1.html > html > body > div.target > img

```
$(".target").slideDown( 'slow', function(){
$(".log").text('Slide Down Transition Complete');
});
});
$("#up").click(function(){
$(".target").slideUp( 'slow', function(){
$(".log").text('Slide Up Transition Complete');
});
});
});
</script>
<style>
p {background-color:#bca; width:200px; border:1px solid green;} img{heigth:50px;width:50px}
</style>
</head>
<body>
<p>Click on any of the buttons</p>
<button id = "up"> Slide Up </button>
<button id = "down"> Slide Down</button>
<div class = "target">
<img    src     =    "fb.png" alt = "jQuery" />
</div>
<div class = "log"></div>
</body>
</html>
```

OUTPUT :



Click on any of the buttons



Click on any of the buttons

Slide Up Transition Complete

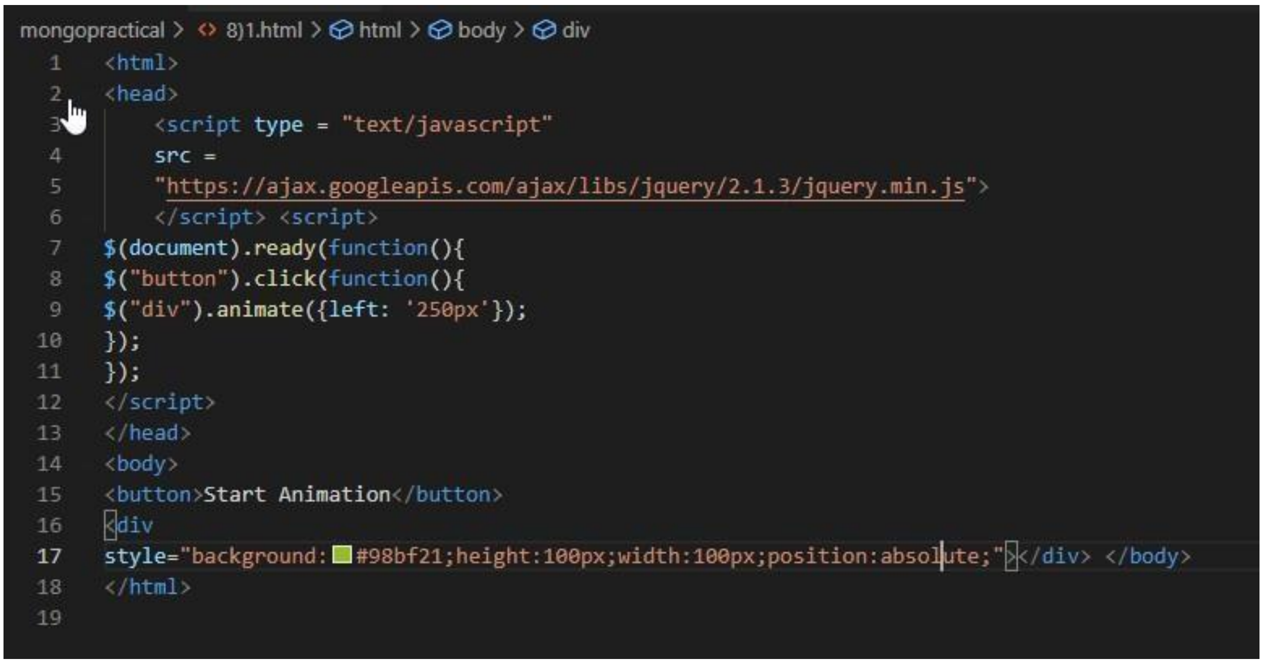
PRACTICAL NO:09

JQUERY ADVANCED

a.jQuery Animation Effect,jQuery Chaining.

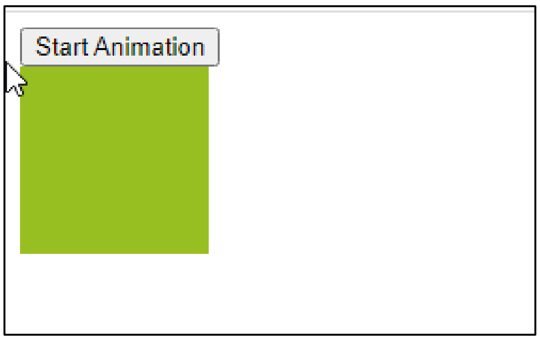
jQuery Animation

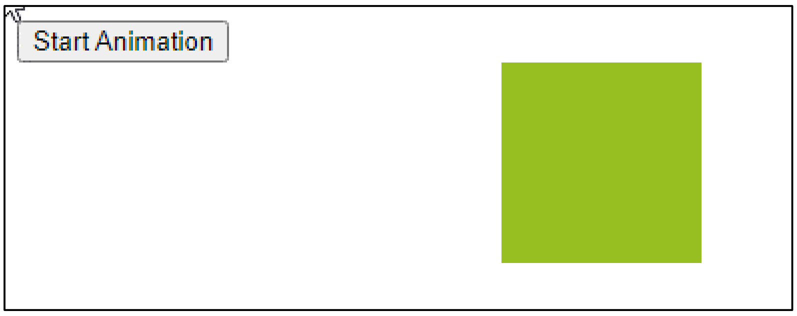
Source Code:



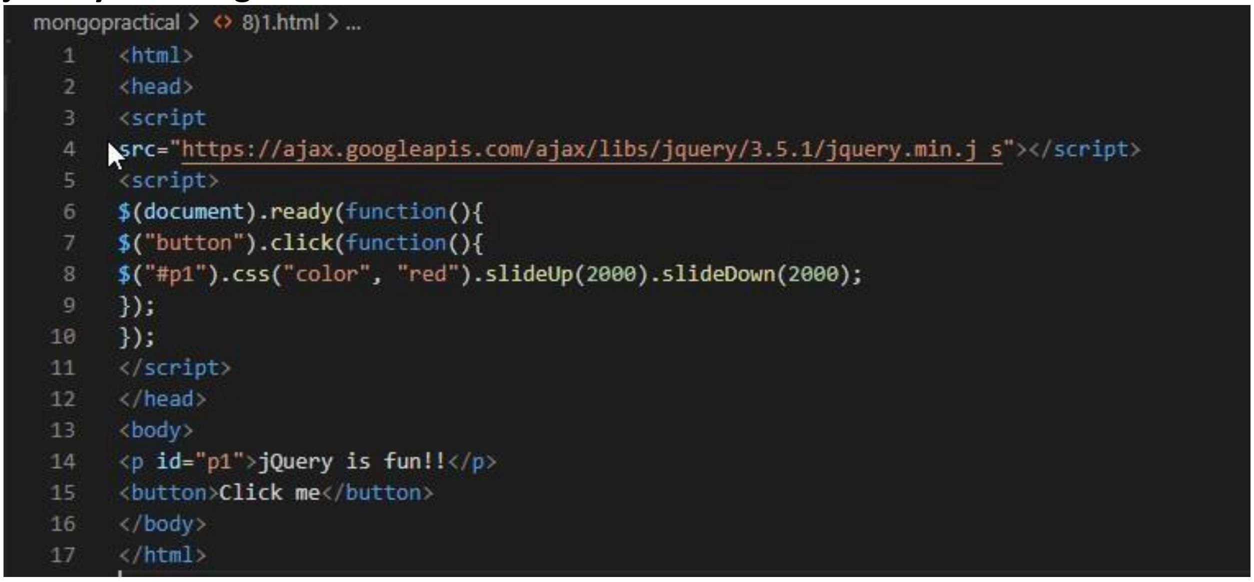
```
mongopractical > <> 8)1.html > html > body > div
<html>
<head>
    <script type = "text/javascript"
    src =
    "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script> <script>
$(document).ready(function(){
$("button").click(function(){
$("div").animate({left: '250px'});
});
});
</script>
</head>
<body>
<button>Start Animation</button>
<div
style="background:#98bf21;height:100px;width:100px;position:absolute;"></div> </body>
</html>
```

OUTPUT :



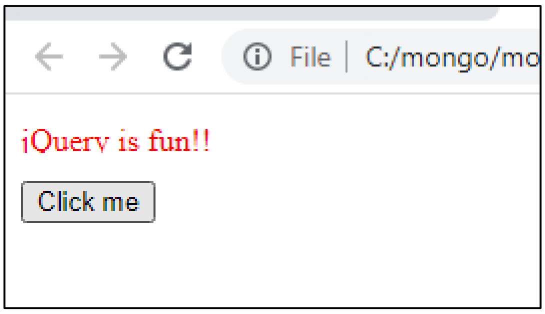

Start Animation

jQuery Chaining - Source Code:



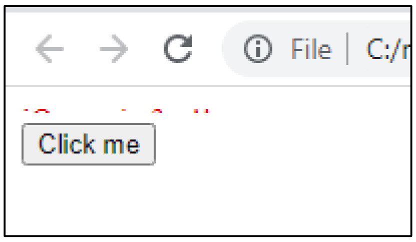
```
mongopractical > <> 8)1.html > ...
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.j s"></script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
});
});
</script>
</head>
<body>
<p id="p1">jQuery is fun!!</p>
<button>Click me</button>
</body>
</html>
```

OUTPUT :

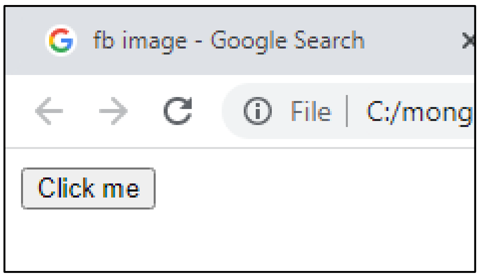


File | C:/mongo/mo

jQuery is fun!!

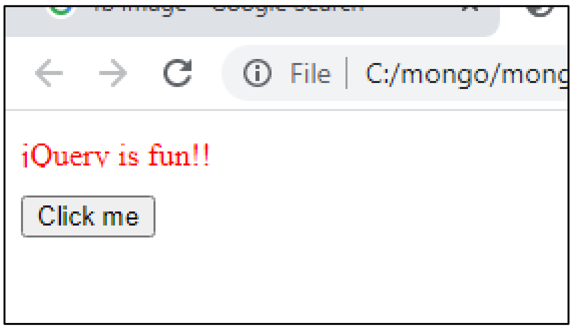


File | C:/r



fb image - Google Search

File | C:/mong



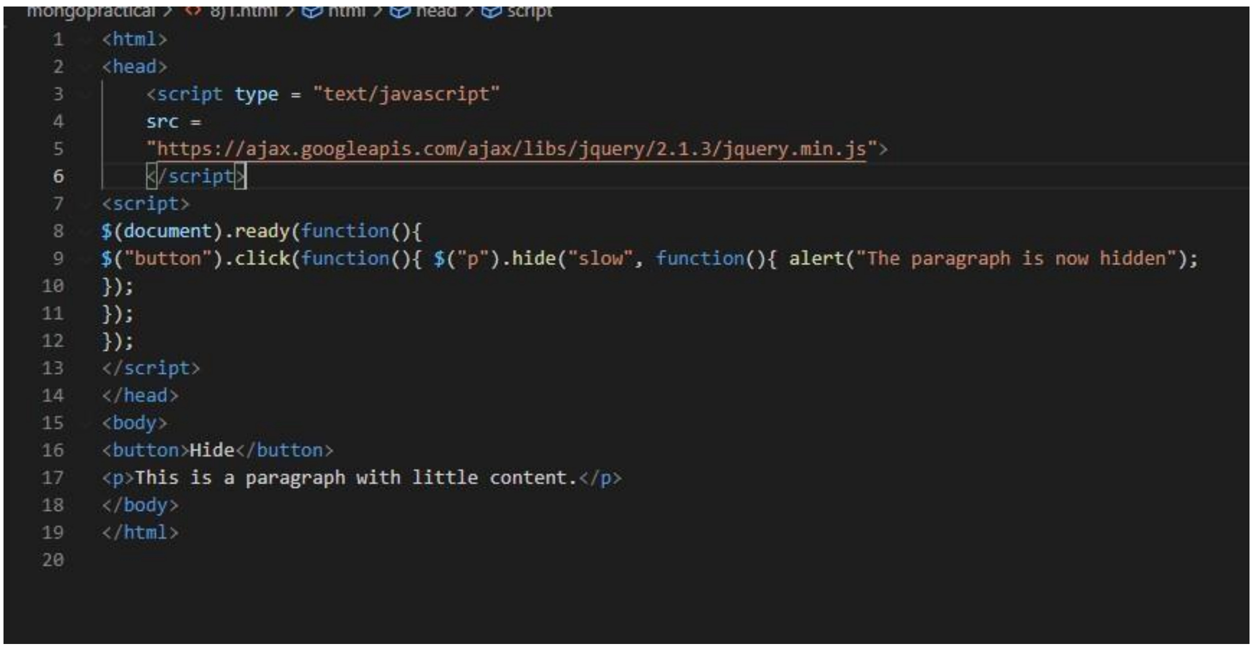
File | C:/mongo/mong

jQuery is fun!!

b) jQuery Callback,jQuery Get and Set Contents

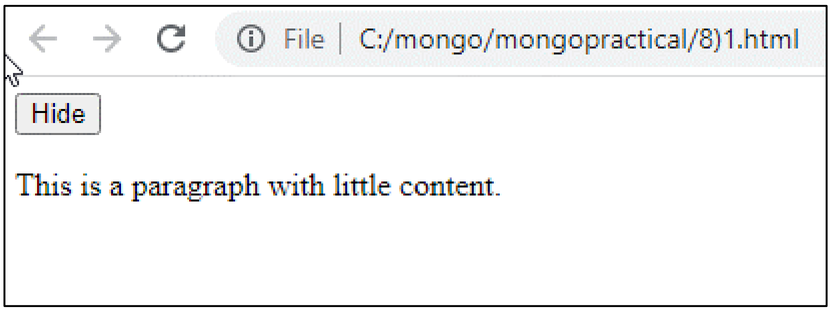
1) jQuery Callback

Source Code:



```
mongopractical > 8)1.html > html > head > script
<html>
<head>
    <script type = "text/javascript"
    src =
    "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>
<script>
$(document).ready(function(){
$("button").click(function(){ $("p").hide("slow", function(){ alert("The paragraph is now hidden");
});
});
});
</script>
</head>
<body>
<button>Hide</button>
<p>This is a paragraph with little content.</p>
</body>
</html>
```

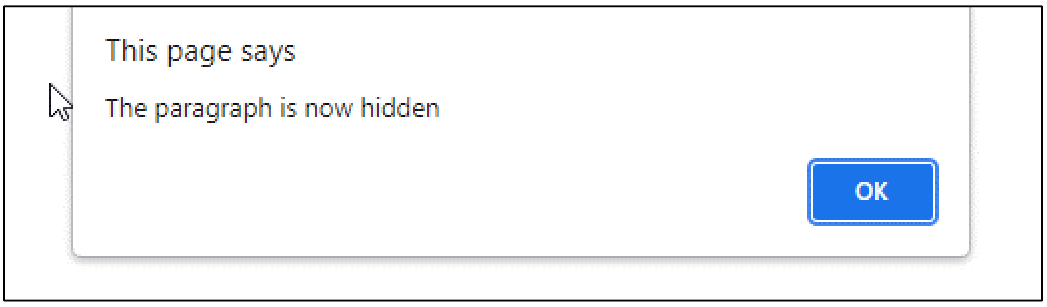
OUTPUT :



File | C:/mongo/mongopractical/8)1.html

Hide

This is a paragraph with little content.



This page says

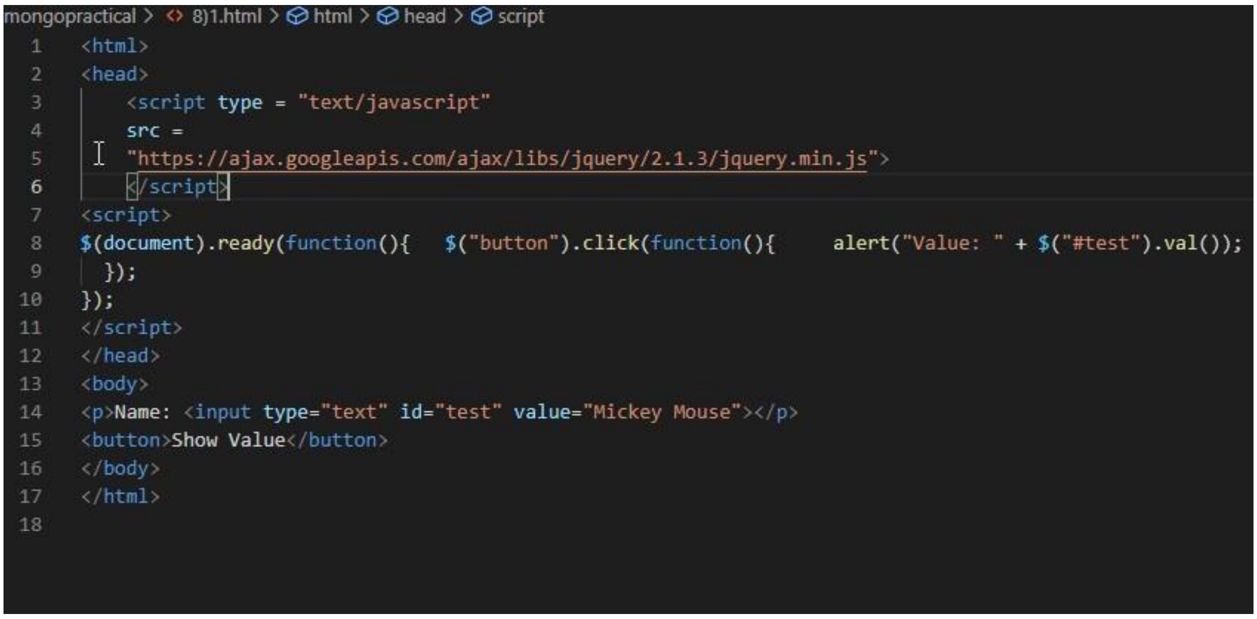
The paragraph is now hidden

OK

2) jQuery Get and Set Contents

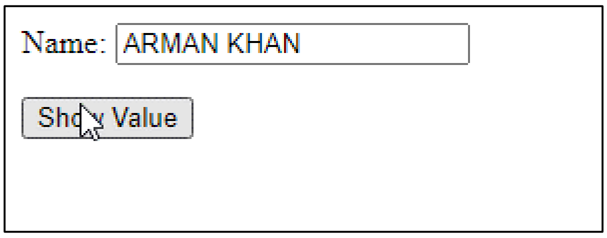
GET Content

Source Code:

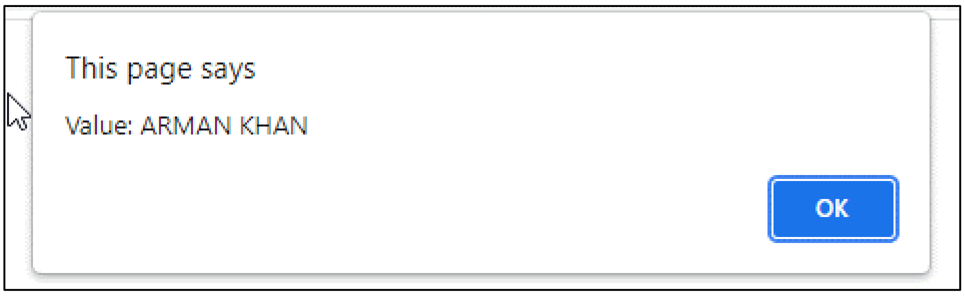


```
mongopractical > <> 8)1.html > html > head > script
<html>
<head>
    <script type = "text/javascript"
    src =
    "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>
<script>
$(document).ready(function(){   $("button").click(function(){     alert("Value: " + $("#test").val());
  });
});
</script>
</head>
<body>
<p>Name: <input type="text" id="test" value="Mickey Mouse"></p>
<button>Show Value</button>
</body>
</html>
```

OUTPUT :



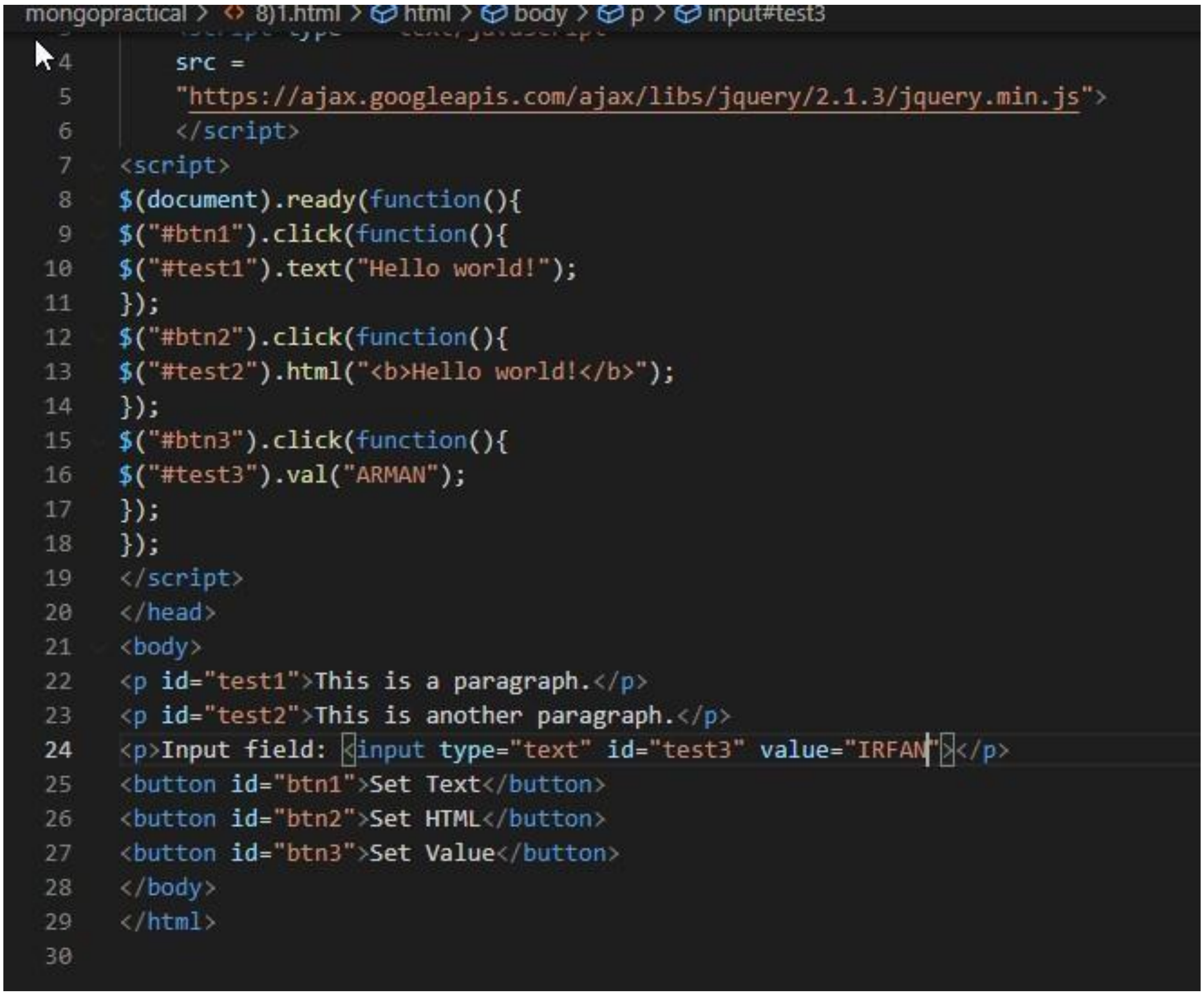
Name: ARMAN KHAN



This page says

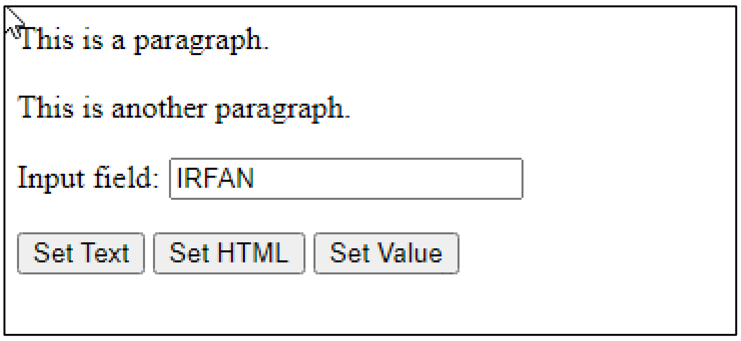
Value: ARMAN KHAN

SET Content Source Code :



```
mongopractical > 8)1.html > html > body > p > input#test3
    src =
    "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>
<script>
$(document).ready(function(){
$("#btn1").click(function(){
$("#test1").text("Hello world!");
});
$("#btn2").click(function(){
$("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
$("#test3").val("ARMAN");
});
});
</script>
</head>
<body>
<p id="test1">This is a paragraph.</p>
<p id="test2">This is another paragraph.</p>
<p>Input field: <input type="text" id="test3" value="IRFAN"></p>
<button id="btn1">Set Text</button>
<button id="btn2">Set HTML</button>
<button id="btn3">Set Value</button>
</body>
</html>
```

OUTPUT :

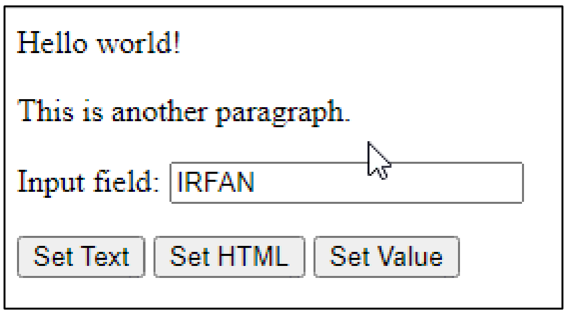


This is a paragraph.

This is another paragraph.

Input field: IRFAN

Set Text Set HTML Set Value

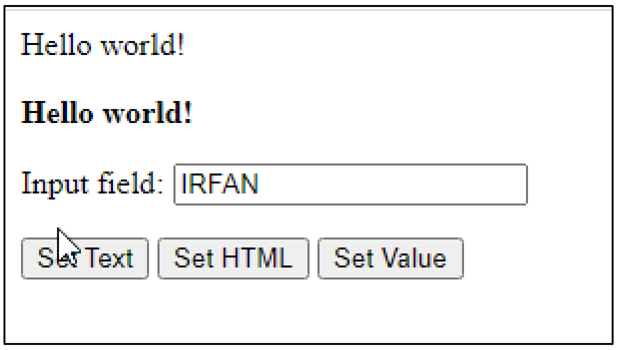


Hello world!

This is another paragraph.

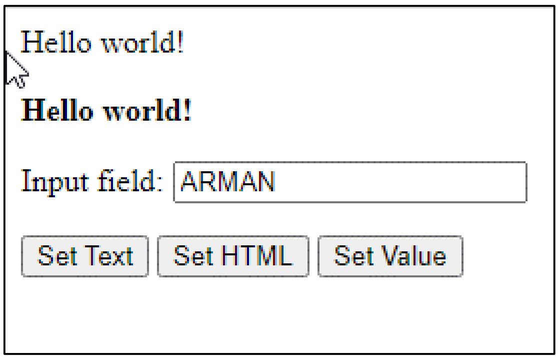
Input field: IRFAN

Set Text Set HTML Set Value


Hello world!

Hello world!

Input field: IRFAN

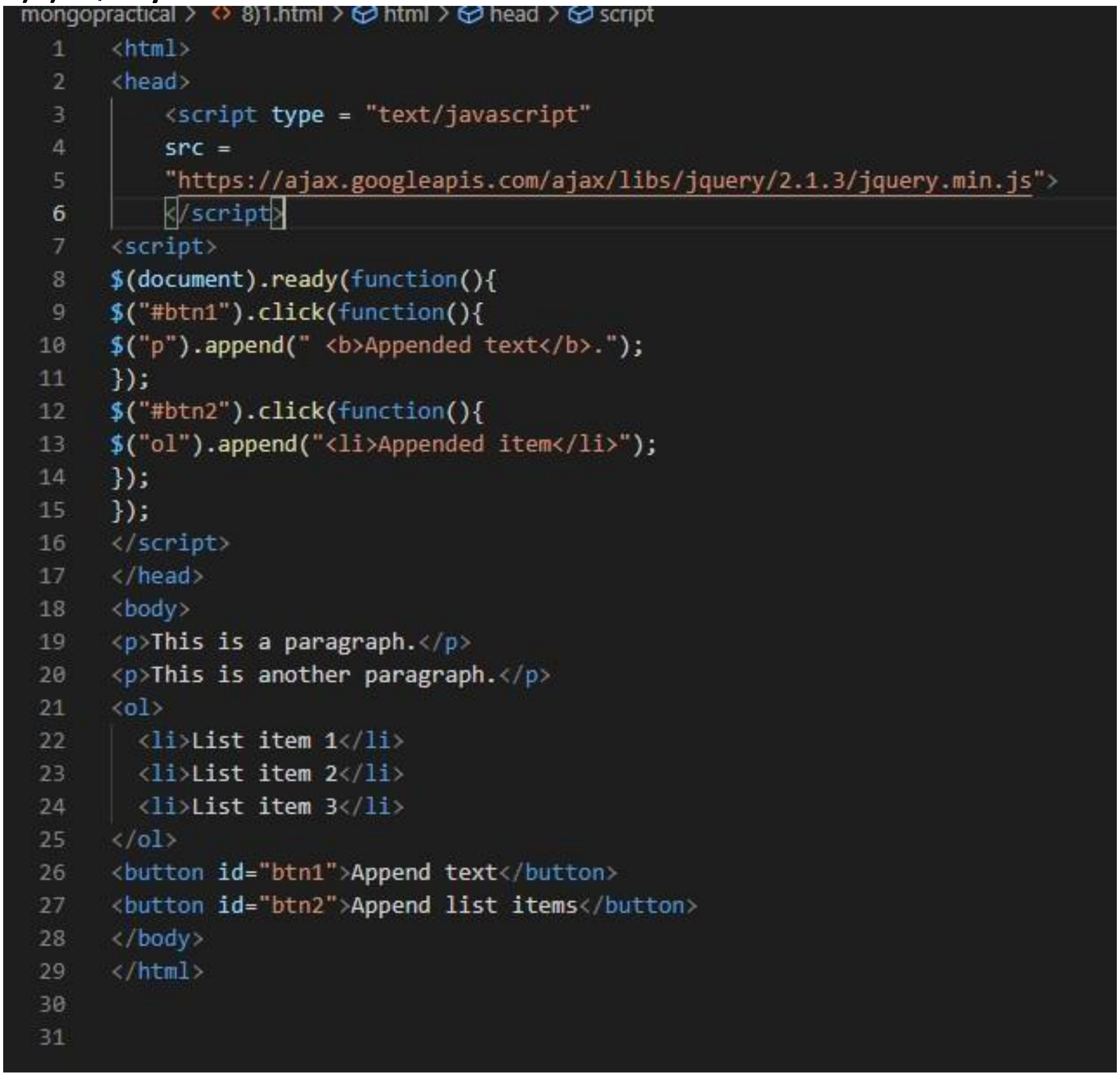

Hello world!

Hello world!

Input field: ARMAN

C. JQuery Insert Content, Remove Content & Attribute

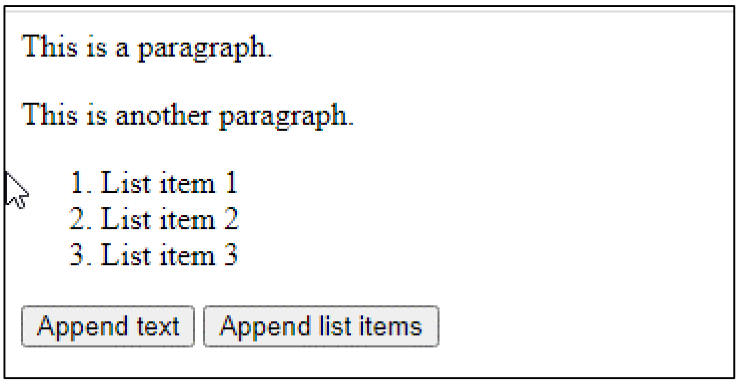
C)1) JQuery Insert Content Source Code :



mongopractical > 8)1.html > html > head > script

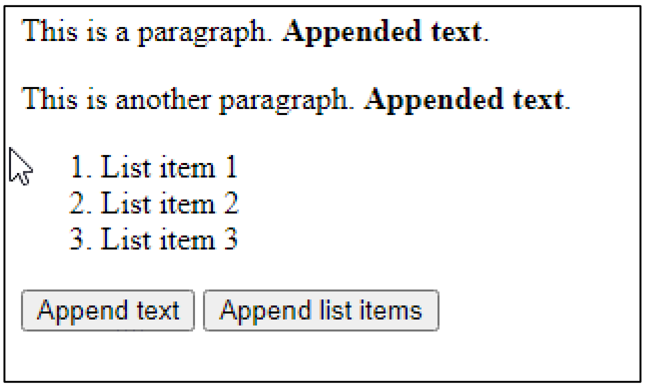
```html
<html>
<head>
    <script type = "text/javascript"
    src =
    "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>
<script>
$(document).ready(function(){
$("#btn1").click(function(){
$("p").append(" <b>Appended text</b>.");
});
$("#btn2").click(function(){
$("ol").append("<li>Appended item</li>");
});
});
</script>
</head>
<body>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<ol>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>
<button id="btn1">Append text</button>
<button id="btn2">Append list items</button>
</body>
</html>
```

OUTPUT :


This is a paragraph.

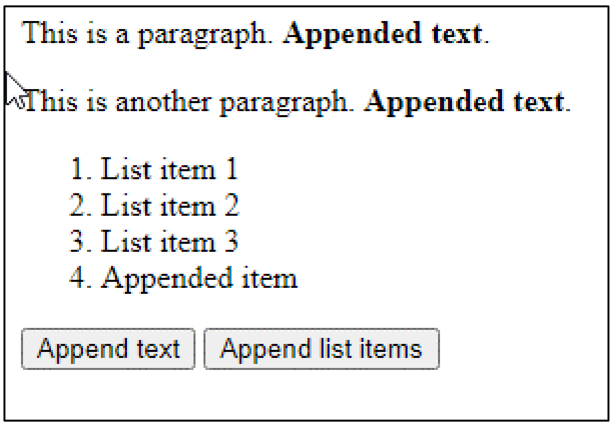
This is another paragraph.

1. List item 1
2. List item 2
3. List item 3


This is a paragraph. **Appended text**.

This is another paragraph. **Appended text**.

1. List item 1
2. List item 2
3. List item 3

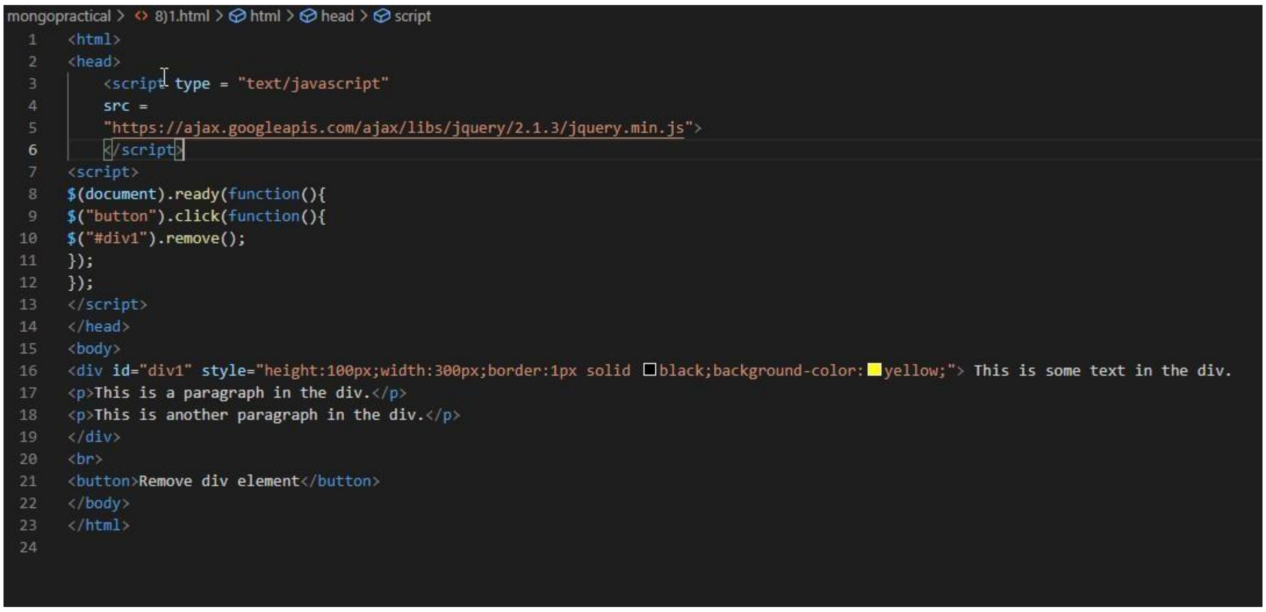

This is a paragraph. **Appended text**.

This is another paragraph. **Appended text**.

1. List item 1
2. List item 2
3. List item 3
4. Appended item

C)2)JQuery Remove Content and attribute

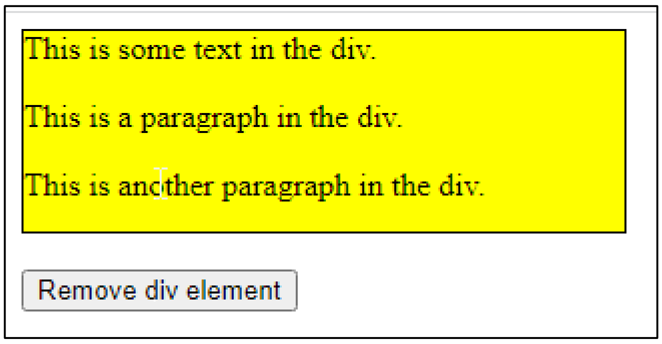
Source Code :



mongopractical > 8)1.html > html > head > script

```
<html>
<head>
    <script type = "text/javascript"
    src =
    "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
    </script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("#div1").remove();
});
});
</script>
</head>
<body>
<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;"> This is some text in the div.
<p>This is a paragraph in the div.</p>
<p>This is another paragraph in the div.</p>
</div>
<br>
<button>Remove div element</button>
</body>
</html>
```

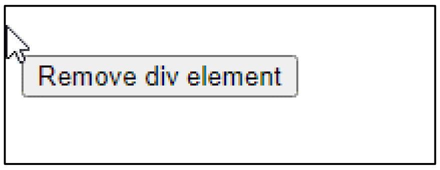
OUTPUT :



This is some text in the div.

This is a paragraph in the div.

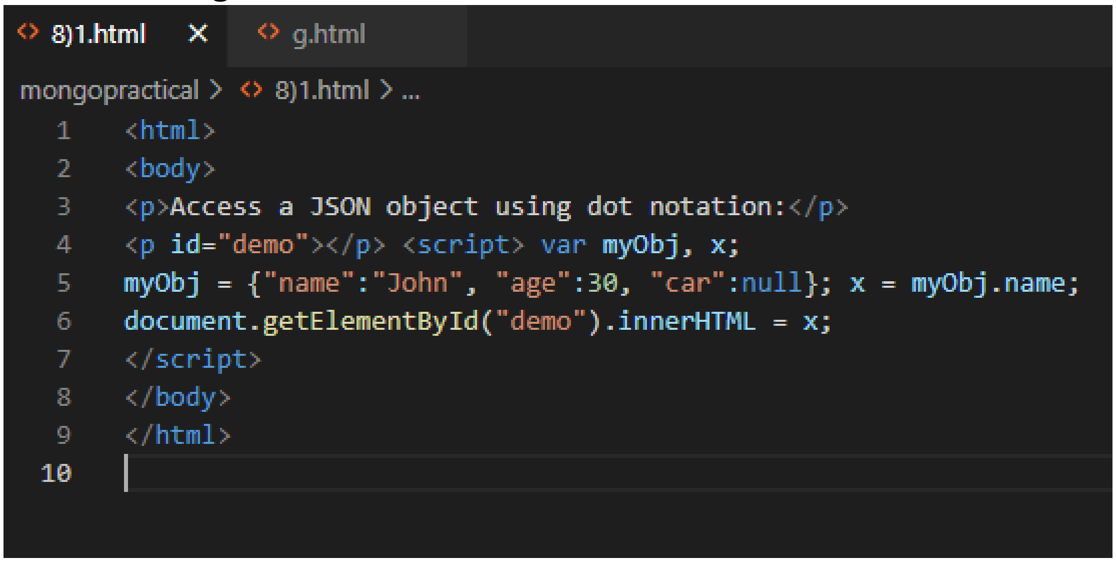
This is another paragraph in the div.



PRACTICAL NO : 10

JSON

A. Creating JSON Source Code :

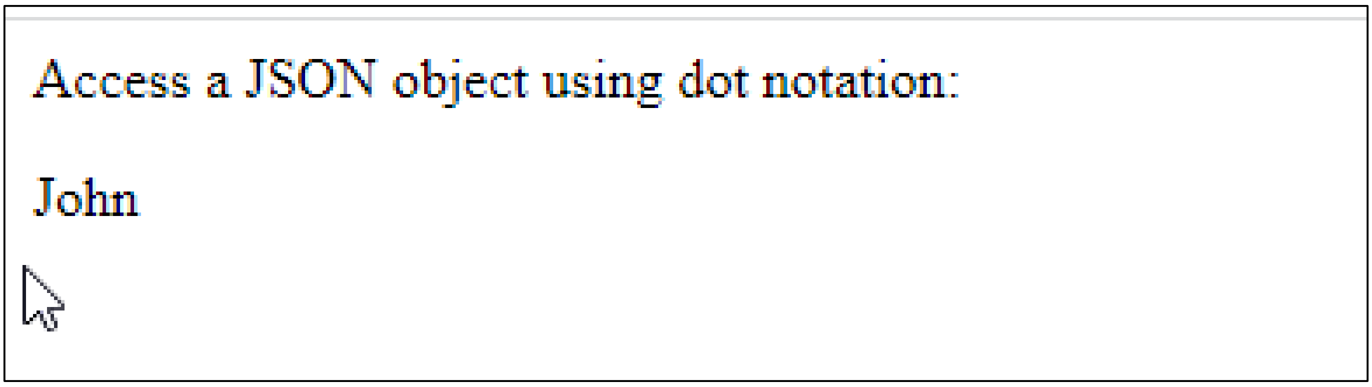


8)1.html × g.html

mongopractical > 8)1.html > ...

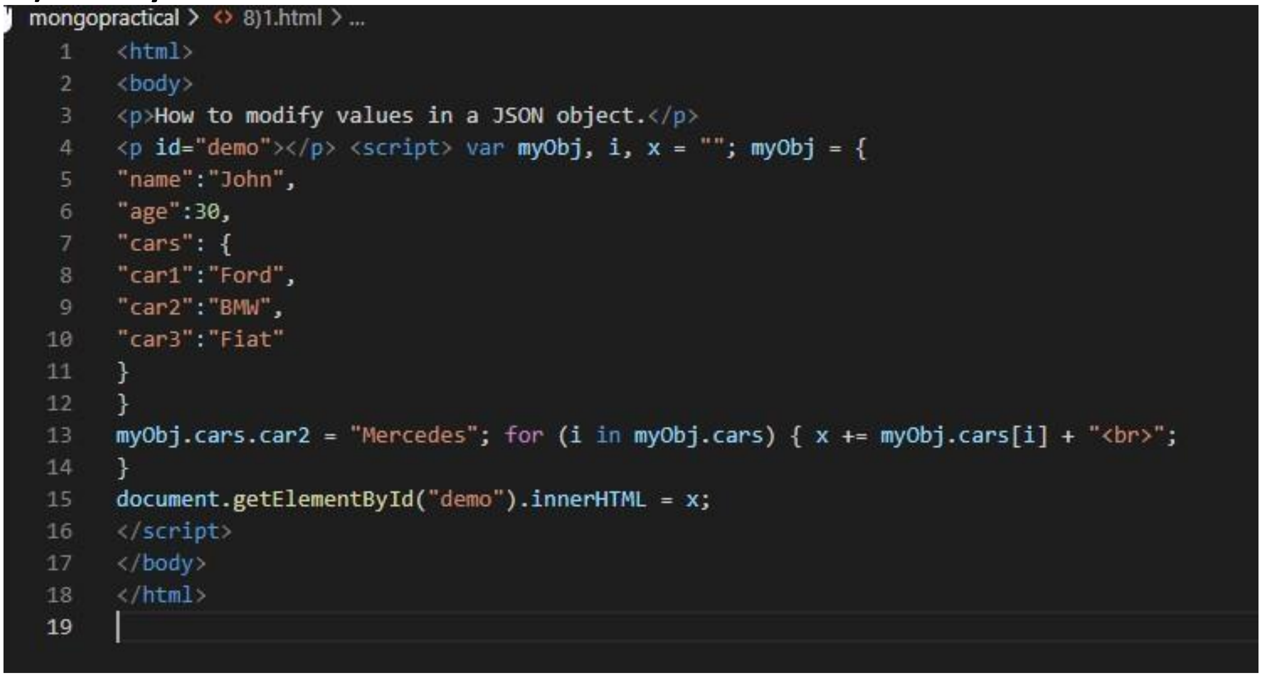
```
<html>
<body>
<p>Access a JSON object using dot notation:</p>
<p id="demo"></p> <script> var myObj, x;
myObj = {"name":"John", "age":30, "car":null}; x = myObj.name;
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

OUTPUT :



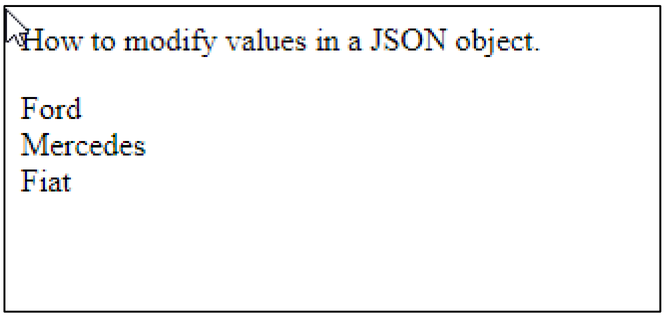
```
Access a JSON object using dot notation:

John
```

B) Modify JSON Source Code :



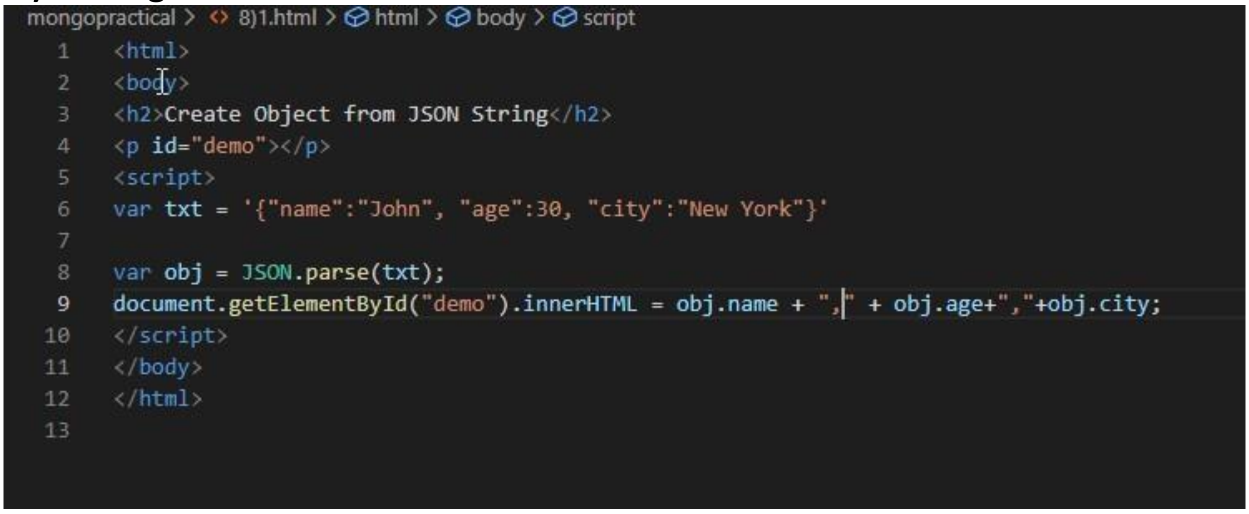
```
<html>
<body>
<p>How to modify values in a JSON object.</p>
<p id="demo"></p> <script> var myObj, i, x = ""; myObj = {
"name":"John",
"age":30,
"cars": {
"car1":"Ford",
"car2":"BMW",
"car3":"Fiat"
}
}
myObj.cars.car2 = "Mercedes"; for (i in myObj.cars) { x += myObj.cars[i] + "<br>";
}
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

OUTPUT



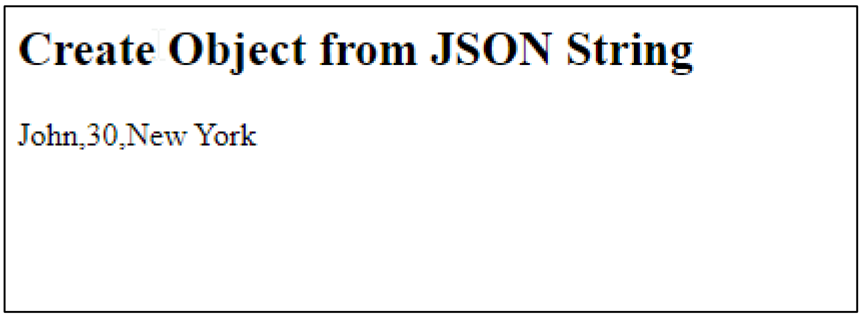
```
How to modify values in a JSON object.

Ford
Mercedes
Fiat
```

C) Parsing JSON Source Code :



```
<html>
<body>
<h2>Create Object from JSON String</h2>
<p id="demo"></p>
<script>
var txt = '{"name":"John", "age":30, "city":"New York"}'

var obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + "," + obj.age+","+obj.city;
</script>
</body>
</html>
```

OUTPUT



```
Create Object from JSON String

John,30,New York
```