This operator is very helpful, whenever we are dealing with large volumes of unstructured data where types are unpredictable.

Syntax-1: Querying for a single type

{field: {$type: <BSONType>}}

We can specifiy either number or alias for the BSON Type.
eg:

{field: {$type: "int"}}
{field: {$type: "string"}}

Syntax-2: Querying for multiple types

{field: {$type: [<BSONType1>, <BSONType2>, <BSONType3>,...]}}

DATATYPES:

Table:
BSON Type-------->Number ---------- >alias
==========
Double-------->1---------->"double"
String-------->2---------- >"string"
Object-------->3---------- >"object"
Array------- >4---------- >"array"
BinaryData---->5 ---------->"binData"
ObjectId------>7 ---------- >"objectId"
Boolean------->8 ---------->"bool"

Date---------->9---------- >"date"

Null--------->10---------- >"null"

32 Bit Integer--------->16 ---------->"int"

64 Bit Integer--------->18 ---------->"long"

Decimal128--------->19 ---------->"decimal"

Q1. What is the difference between int and long?

----

int --->32 bits integer value

long --->64 bits integer value

Q2. What is the difference between double and decimal?

----

double --->64 bits floating point value

decimal --->128 bits floating point value

Note:

$type supports "number" alias, which will match the following BSON Types.

int

long

double

decimal

Case Study:

```
db.phonebook.insertOne({_id: 1, name: "Sunny", phoneNumber: "9292929292"})
db.phonebook.insertOne({_id: 2, name: "Bunny", phoneNumber: 8896979797})
db.phonebook.insertOne({_id: 3, name: "Chinny", phoneNumber: NumberLong(9898989898) })
db.phonebook.insertOne({_id: 4, name: "Vinny", phoneNumber: NumberInt(9246212143)})
db.phonebook.insertOne({_id: 5, name: "Pinny", phoneNumber: ["8885252627", 8096969696]})
```

Every number is bydefault treated as double type in MongoDB.

"9292929292" --->string type
8896979797 ----->double type
NumberLong(9898989898) --->long type
NumberInt(9246212143)--->int type

Q1. Select all documents where phoneNumber value is of string type?

```
> db.phonebook.find({phoneNumber: {$type: "string"}}).pretty()
> db.phonebook.find({phoneNumber: {$type: 2}}).pretty()

> db.phonebook.find({phoneNumber: {$type: 2}}).pretty()
{ "_id" : 1, "name" : "Sunny", "phoneNumber" : "9292929292" }
{
        "_id" : 5,
        "name" : "Pinny",
        "phoneNumber" : [
                "8885252627",
                8096969696
        ]
}
```

Q2. Select all documents where phoneNumber value is of double type?

```
> db.phonebook.find({phoneNumber: {$type: "double"}}).pretty()
> db.phonebook.find({phoneNumber: {$type: 1}}).pretty()

> db.phonebook.find({phoneNumber: {$type: 1}}).pretty()
{ "_id" : 2, "name" : "Bunny", "phoneNumber" : 8896979797 }
{
        "_id" : 5,
```

```
        "name" : "Pinny",
        "phoneNumber" : [
                "8885252627",
                8096969696
        ]
}
```

Q3. Select all documents where phoneNumber value is of int type?

```
> db.phonebook.find({phoneNumber: {$type: "int"}}).pretty()
> db.phonebook.find({phoneNumber: {$type: 16}}).pretty()


> db.phonebook.find({phoneNumber: {$type: 16}}).pretty()
{ "_id" : 4, "name" : "Vinny", "phoneNumber" : 656277551 }
```

Note: NumberInt(9246212143) -->656277551
9246212143 cannot be accomodated in 32 bits. Hence some loss of information.

Q4. Select all documents where phoneNumber value is of long type?

```
> db.phonebook.find({phoneNumber: {$type: "long"}}).pretty()
> db.phonebook.find({phoneNumber: {$type: 18}}).pretty()

> db.phonebook.find({phoneNumber: {$type: 18}}).pretty()
{ "_id" : 3, "name" : "Chinny", "phoneNumber" :
NumberLong("9898989898") }
```

Q5. Select all documents where phoneNumber value is of number type?

```
> db.phonebook.find({phoneNumber: {$type: "number"}}).pretty()

{ "_id" : 2, "name" : "Bunny", "phoneNumber" : 8896979797 }
{ "_id" : 3, "name" : "Chinny", "phoneNumber" :
NumberLong("9898989898") }
{ "_id" : 4, "name" : "Vinny", "phoneNumber" : 656277551 }
{
        "_id" : 5,
        "name" : "Pinny",
        "phoneNumber" : [
                "8885252627",
                8096969696
        ]
}
```

Q6. Querying by multiple data types

Select all documents where phoneNumber value is of either string or double.

```
> db.phonebook.find({phoneNumber: {$type: ["string",
"double"]}}).pretty()
> db.phonebook.find({phoneNumber: {$type: [2, 1]}}).pretty()

> db.phonebook.find({phoneNumber: {$type: [2, 1]}}).pretty()
{ "_id" : 1, "name" : "Sunny", "phoneNumber" : "9292929292" }
{ "_id" : 2, "name" : "Bunny", "phoneNumber" : 8896979797 }
{
        "_id" : 5,
        "name" : "Pinny",
        "phoneNumber" : [
                "8885252627",
```

```
            8096969696
        ]
}
```

comparison operators: $gt,$gte,$lt,$lte,$eq,$ne,$in,$nin
logical operators: $or, $nor, $and, $not
element query operators: $exists, $type

Evaluation Query Operators:

The operators which can be used for evaluation purposes are called Evaluation Query Operators.

1. $expr 2. $regex 3. $mod 4. $jsonSchema 5. $text 6. $where

1.$expr operator:

---

expr means expression.
Evaluate expression and select documents which satisfy that expression.
Syntax:
{ $expr: {<expression>}}

It is very helpful to compare two field values within the same document.