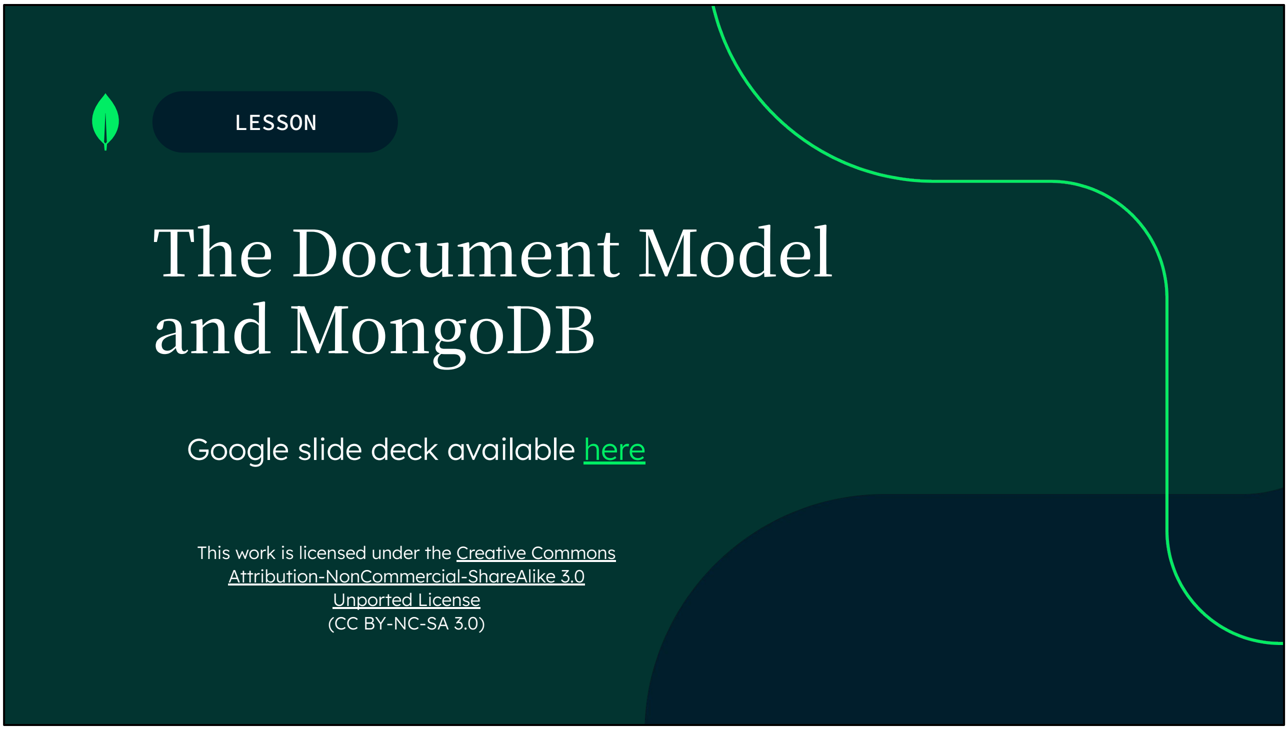


LESSON

The Document Model and MongoDB

Google slide deck available here

Overview

Learning Objectives

At the end of this course, learners will be able to:

- Describe the four key features of the document model.
- Identify and apply the correct syntax and structure used in the document model.
- Explain how data is stored using BSON.
- Understand the difference between BSON and JSON in regards to storing data in MongoDB’s document model.
- Define a collection in the document model.

Suggested Uses

- Lecture for one hour class
- Handouts / asynchronous learning
- Supplemental reading material - read on your own / not part of formal teaching
- Complement to University course Introduction to MongoDB

This lesson is a part of the course Introduction to Modern Databases with MongoDB.

At a Glance

Length:
45 minutes

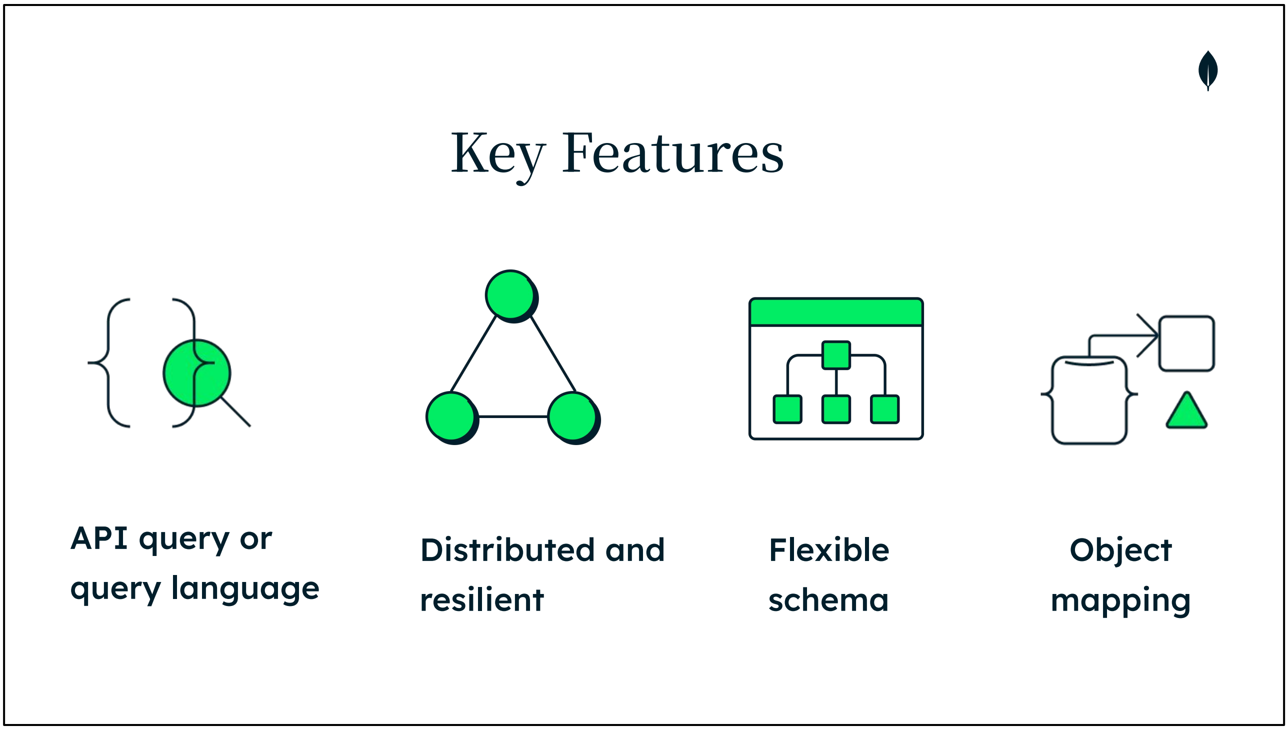
Level:
Foundational

Prerequisites:
Non-Relational Database Types

Share your feedback: We hope these curriculum materials will be a valuable resource for you and your learners. Let us know how the materials work for you, what we can improve on, and how MongoDB for Academia can support you via our brief feedback form.

MongoDB for Academia: MongoDB for Academia offers resources for educators and students to support teaching and learning MongoDB. Check out our educator resources and join the Educator Community. Students can receive $50 in Atlas credits and free certification through the GitHub Student Developer Pack.

Last Update: March 2025



Key Features

API query or query language

Distributed and resilient

Flexible schema

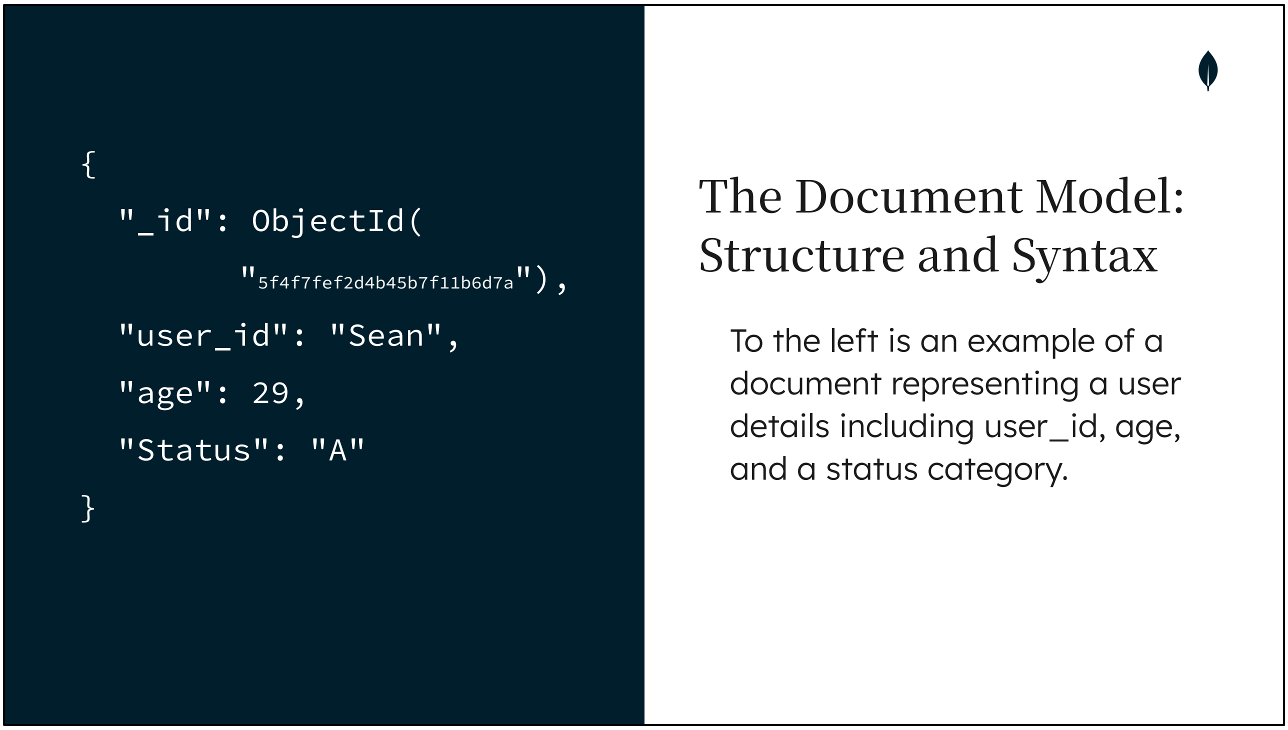
Object mapping

Querying through an API or query language: Document databases have an API or query language that allows developers to execute the CRUD operations on the database. Developers have the ability to query for documents based on unique identifiers or "field values."

Distributed and resilient: Document databases are distributed, which allows for horizontal scaling (typically cheaper than vertical scaling) which distributes the data across multiple machines rather than making one machine bigger as the data increases. This system also allows for data to achieve high availability and resiliency as the data lives in replica sets which creates redundancy, so if one machine fails the secondary machine will take over and keep the data alive. This system is also referred to as sharding.

Object mapping: Documents easily map to objects, the most frequently used data structure in the most popular programming languages. This allows developers to rapidly develop their applications as it is an intuitive process.

Flexible schema: Document databases have a flexible schema, meaning that not all documents in a collection need to have the same fields. Note that some document databases support schema validation, so the schema can be both mandatory or defined.

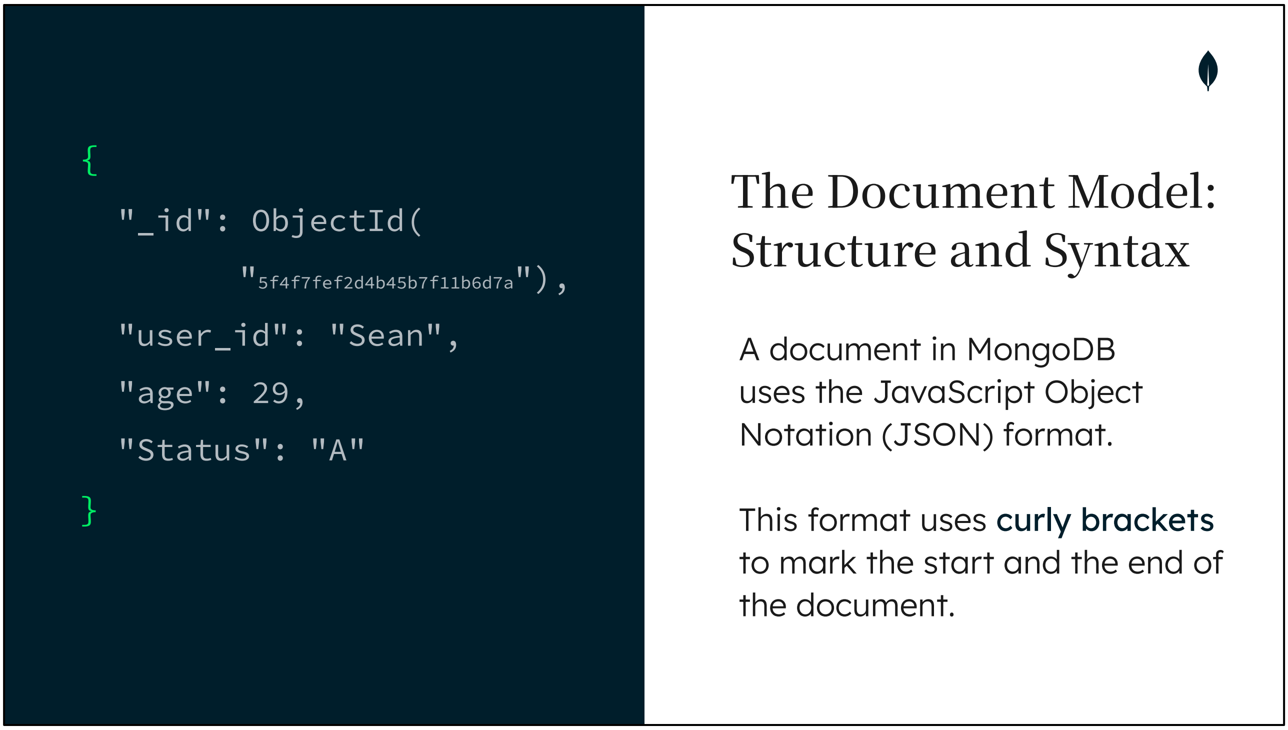


```
{
  "_id": ObjectId(
         "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Sean",
  "age": 29,
  "Status": "A"
}
```

The Document Model: Structure and Syntax

To the left is an example of a document representing a user details including user_id, age, and a status category.

In order to better understand a document, let's take an example in MongoDB. This document represents a user, their id in the system, their age, and their status. You can also note the "_id" field which holds the ObjectID for the document. The _id is used as a primary key; its value must be unique in the collection, it is immutable, and may be of any type other than an array. In this example, it uses a number but typically these will be automatically generated ObjectIDs. An ObjectID is a small, likely unique, fast to generate, and ordered 12 byte value. An ObjectID's 12 bytes consist of a 4-byte timestamp value, representing the ObjectID creation time, measured in seconds since the Unix epoch, a 5-byte random value, and a 3-byte incrementing counter, initialized to a random value.



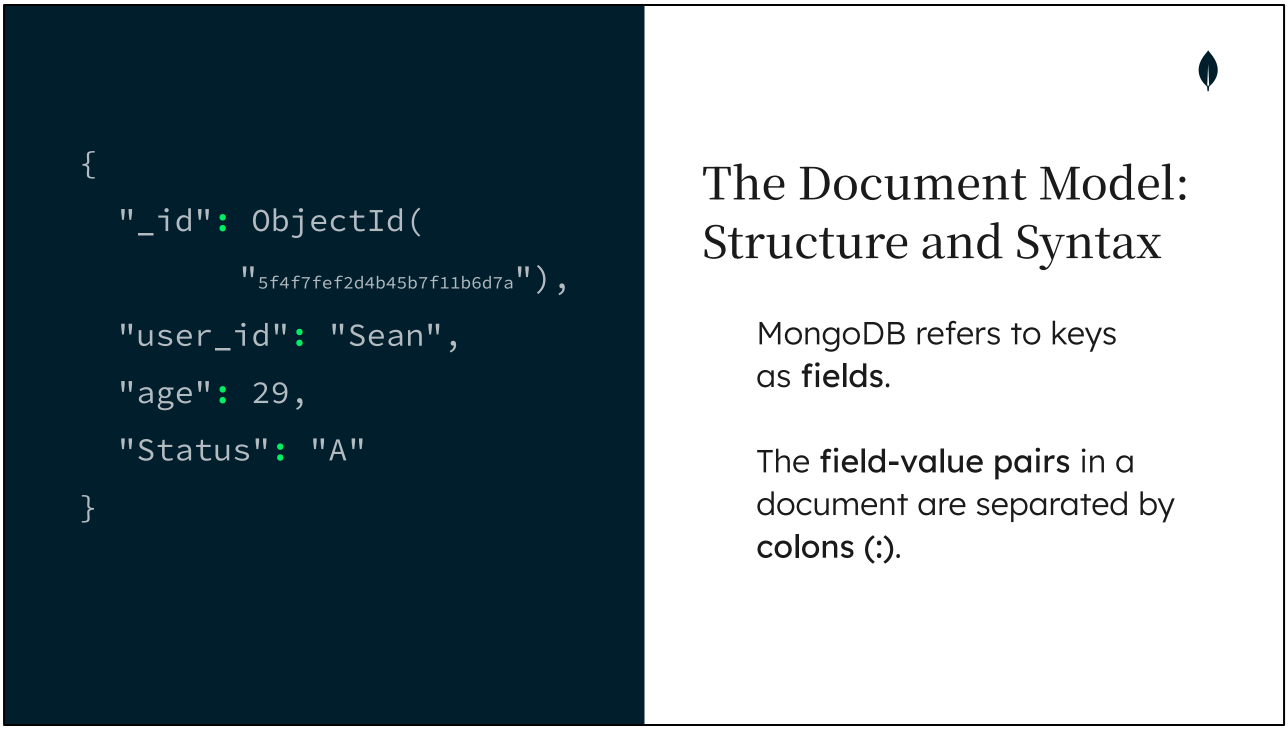
```
{
  "_id": ObjectId(
         "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Sean",
  "age": 29,
  "Status": "A"
}
```

The Document Model: Structure and Syntax

A document in MongoDB uses the JavaScript Object Notation (JSON) format.

This format uses **curly brackets** to mark the start and the end of the document.

In order to better understand a document, let’s take an example in MongoDB. This document represents a user, their id in the system, their age, and their status. You can also note the “_id” field which is holds the ObjectID for the document. The _id is used as a primary key; its value must be unique in the collection, it is immutable, and may be of any type other than an array. In this example, it uses an number but typically these will be automatically generated ObjectIDs. An ObjectID is a small, likely unique, fast to generate, and ordered 12 byte value. An ObjectID 12 bytes consist of a 4-byte timestamp value, representing the ObjectId creation time, measured in seconds since the Unix epoch, a 5-byte random value, and a 3-byte incrementing counter, initialized to a random value.



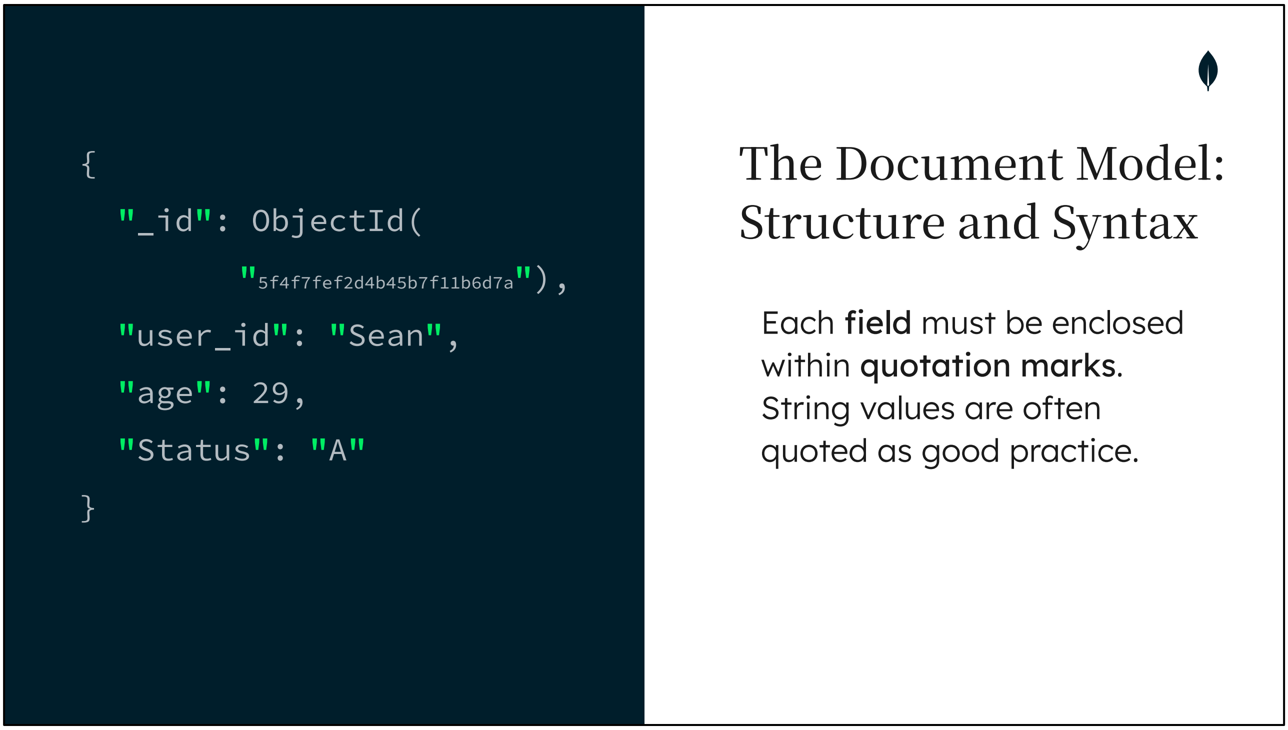
```
{
  "_id": ObjectId(
        "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Sean",
  "age": 29,
  "Status": "A"
}
```

The Document Model: Structure and Syntax

MongoDB refers to keys as **fields**.

The **field-value pairs** in a document are separated by **colons (:)**.

In order to better understand a document, let’s take an example in MongoDB. This document represents a user, their id in the system, their age, and their status. You can also note the “_id” field which is holds the ObjectID for the document. The _id is used as a primary key; its value must be unique in the collection, it is immutable, and may be of any type other than an array. In this example, it uses an number but typically these will be automatically generated ObjectIDs. An ObjectID is a small, likely unique, fast to generate, and ordered 12 byte value. An ObjectID 12 bytes consist of a 4-byte timestamp value, representing the ObjectId creation time, measured in seconds since the Unix epoch, a 5-byte random value, and a 3-byte incrementing counter, initialized to a random value.

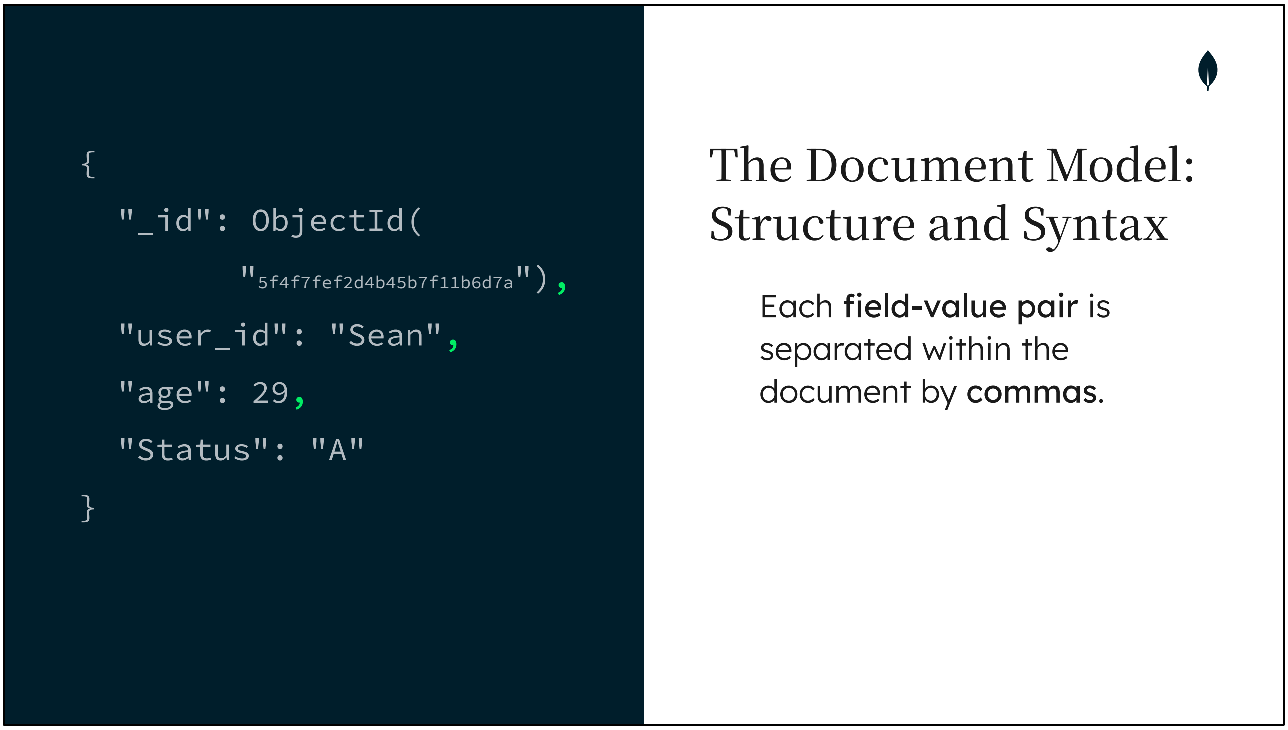


```
{
  "_id": ObjectId(
         "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Sean",
  "age": 29,
  "Status": "A"
}
```

The Document Model: Structure and Syntax

Each **field** must be enclosed within **quotation marks.** String values are often quoted as good practice.

Each field in a MongoDB document must be enclosed within quotation marks. String values are often quoted as good practice.



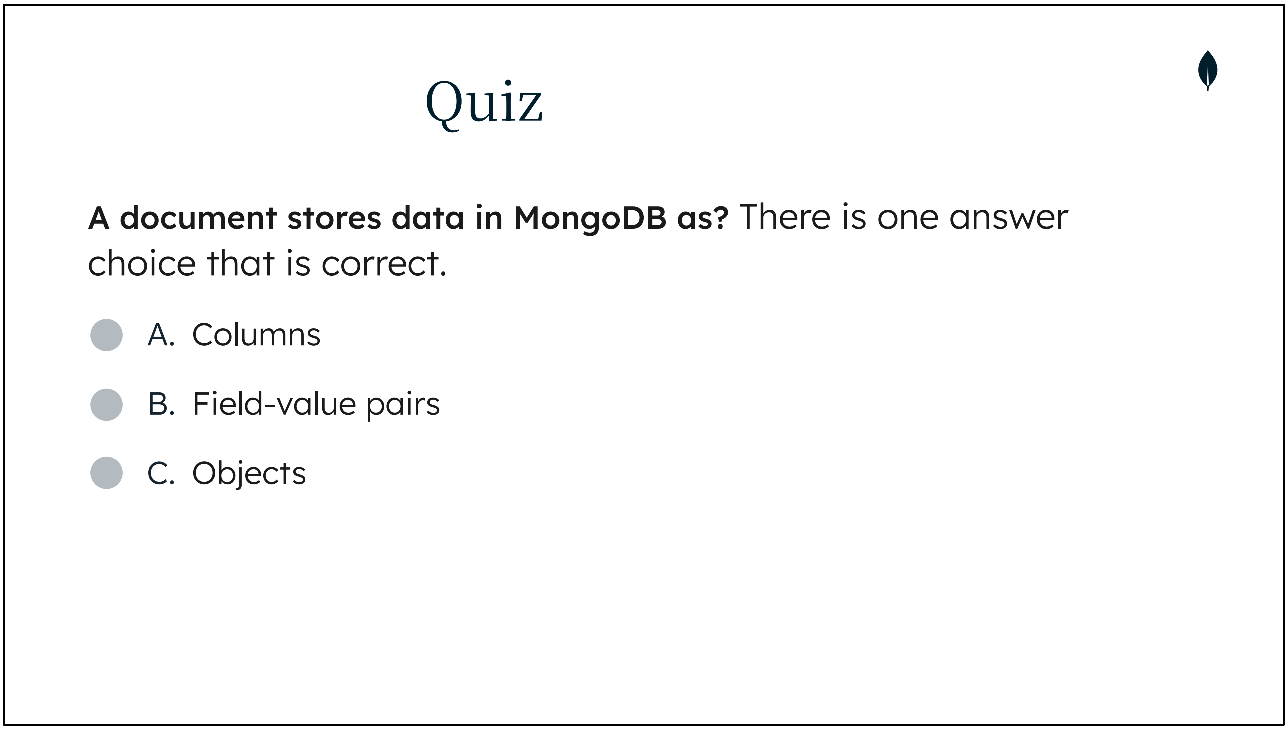
```
{
  "_id": ObjectId(
         "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Sean",
  "age": 29,
  "Status": "A"
}
```

The Document Model: Structure and Syntax

Each **field-value pair** is separated within the document by **commas**.

Each of the field-value pairs in the document are separated by a comma from the next record. The final field-value pair doesn't require a comma as the final curly brace indicates the end of the document.

Quiz



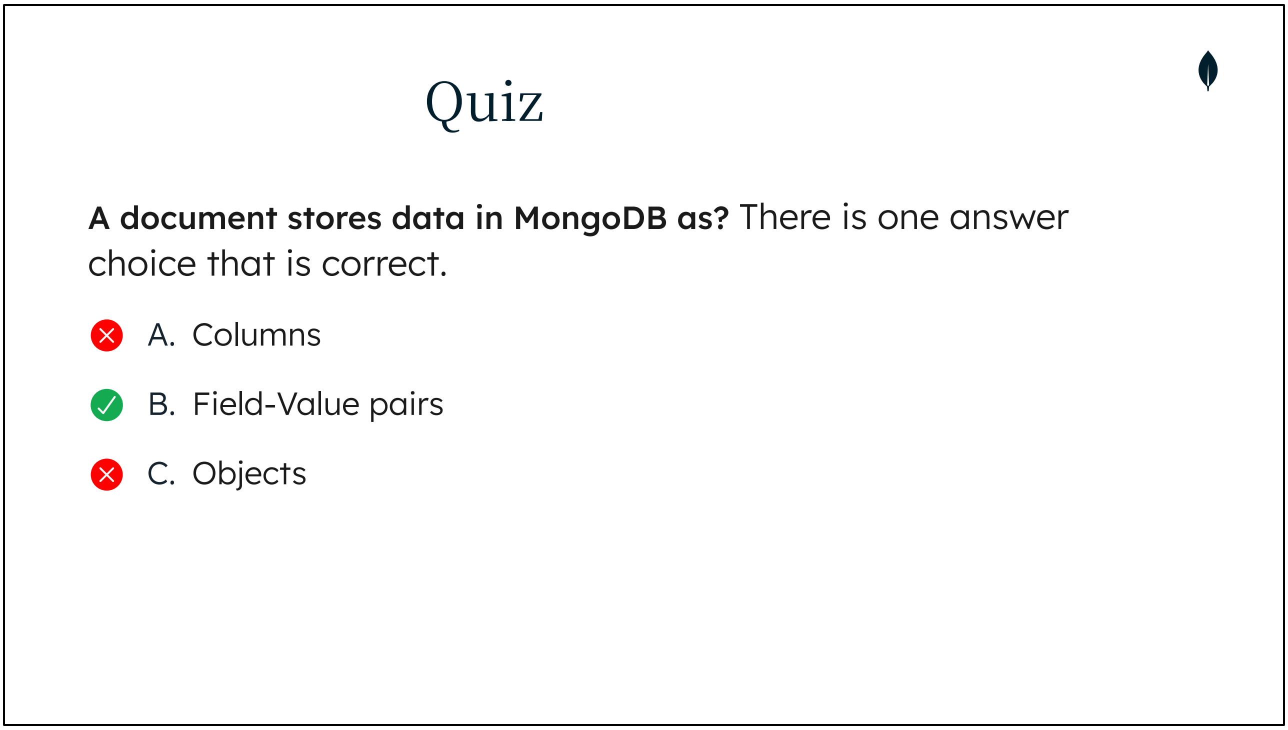
Quiz

A document stores data in MongoDB as? There is one answer choice that is correct.

○ A. Columns

○ B. Field-value pairs

○ C. Objects



Quiz

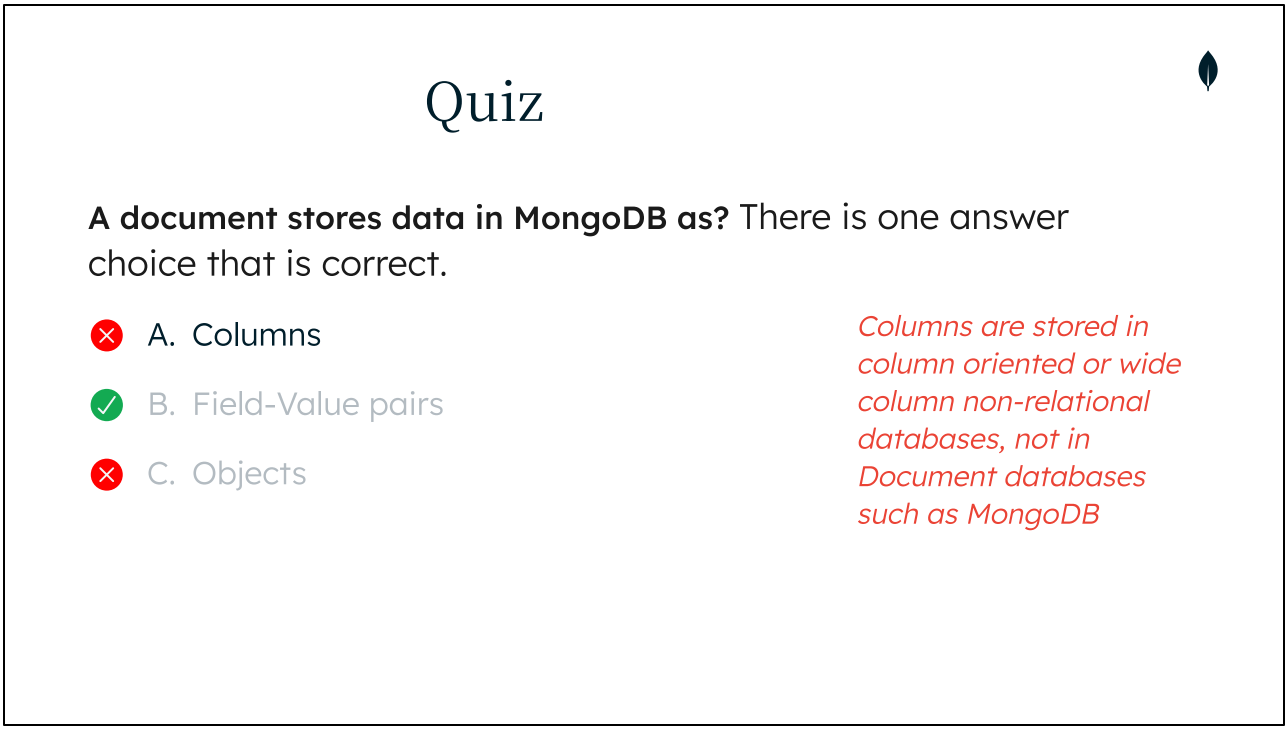
A document stores data in MongoDB as? There is one answer choice that is correct.

✗ A. Columns

✓ B. Field-Value pairs

✗ C. Objects

INCORRECT: Columns - Columns are stored in column oriented or wide column non-relational databases, not in Document databases such as MongoDB.
CORRECT: Field-Value pairs - MongoDB stores field-value pairs where the value can be one of several data types including a sub-object.
INCORRECT: Objects - MongoDB does not store data as object, rather it is stored as a field-value pair within a object.



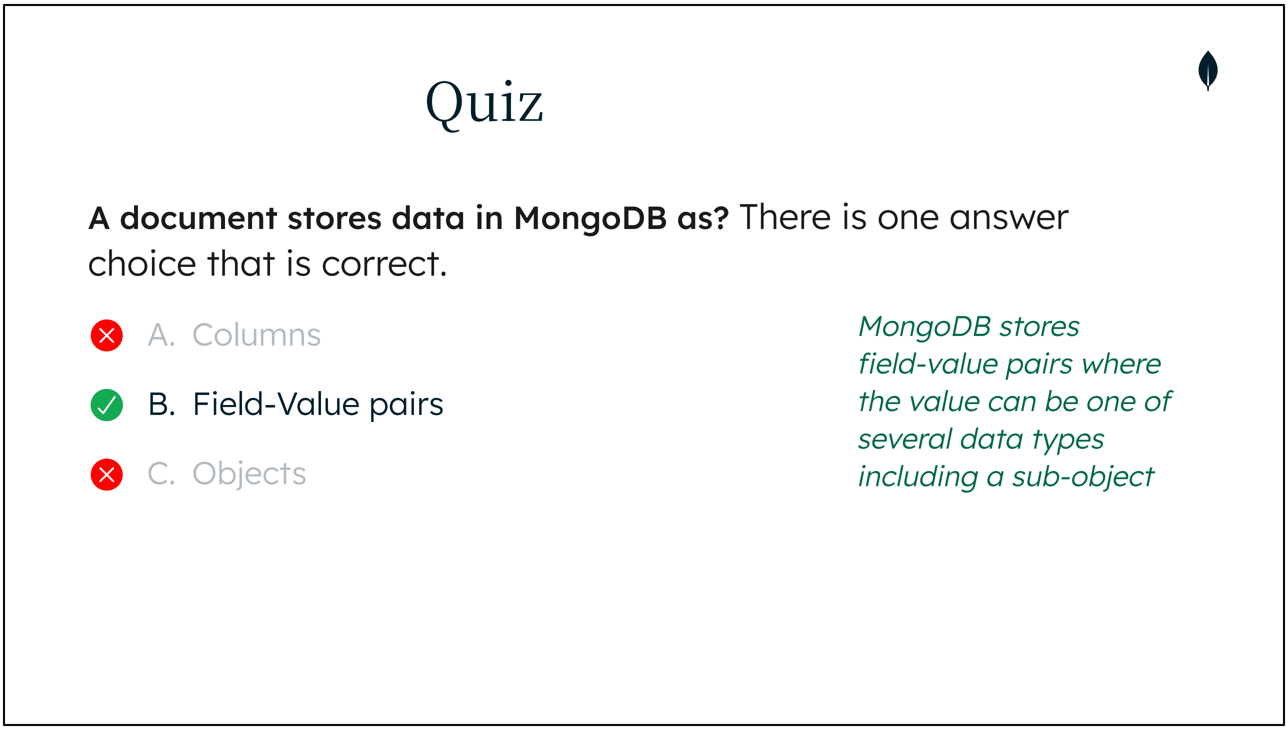
Quiz

A document stores data in MongoDB as? There is one answer choice that is correct.

- ✗ A. Columns
- ✓ B. Field-Value pairs
- ✗ C. Objects

Columns are stored in column oriented or wide column non-relational databases, not in Document databases such as MongoDB

INCORRECT: Columns - Columns are stored in column oriented or wide column non-relational databases, not in Document databases such as MongoDB.



Quiz

A document stores data in MongoDB as? There is one answer choice that is correct.

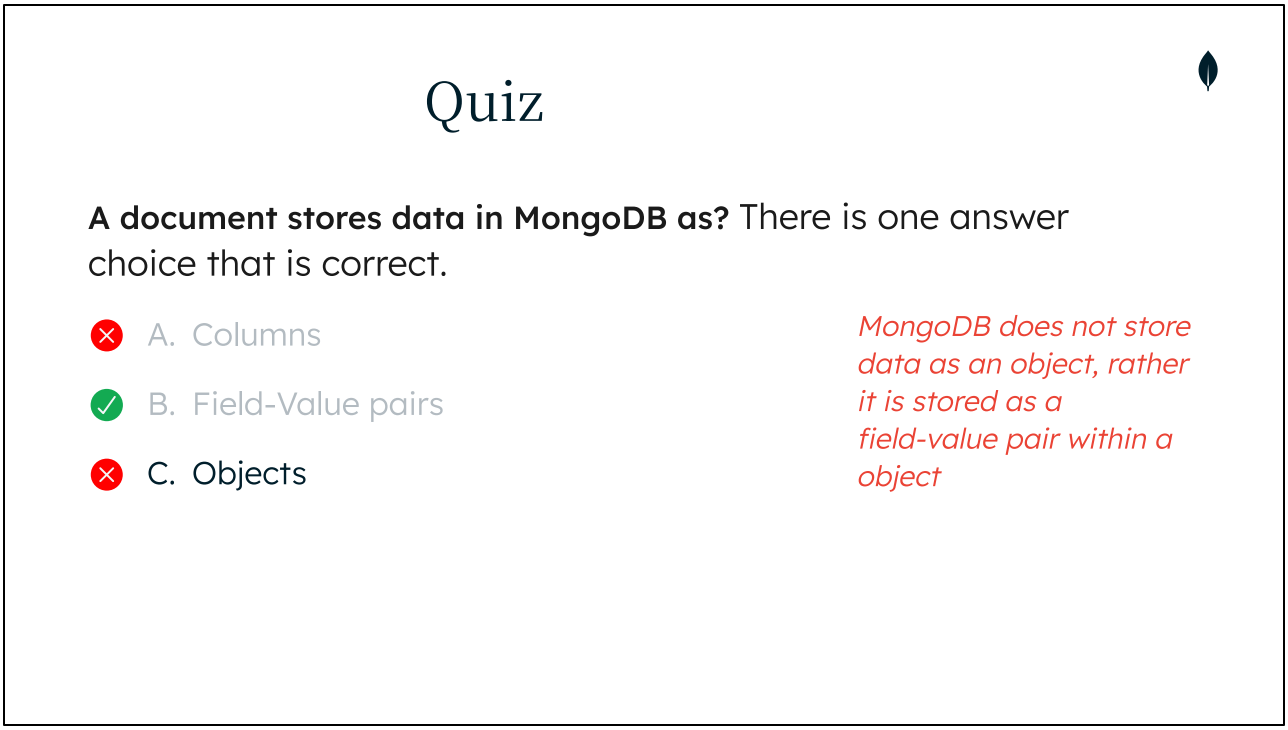
✗ A. Columns

✓ B. Field-Value pairs

✗ C. Objects

MongoDB stores field-value pairs where the value can be one of several data types including a sub-object

CORRECT: Field-Value pairs - MongoDB stores field-value pairs where the value can be one of several data types including a sub-object.



Quiz

A document stores data in MongoDB as? There is one answer choice that is correct.

- ✗ A. Columns
- ✓ B. Field-Value pairs
- ✗ C. Objects

MongoDB does not store data as an object, rather it is stored as a field-value pair within a object

INCORRECT: Objects - MongoDB does not store data as object, rather it is stored as a field-value pair within a object.

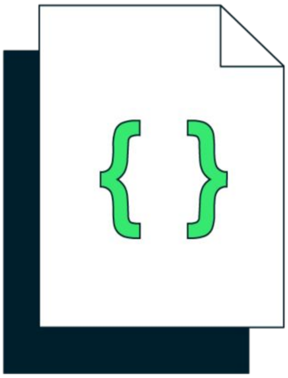
JSON and BSON

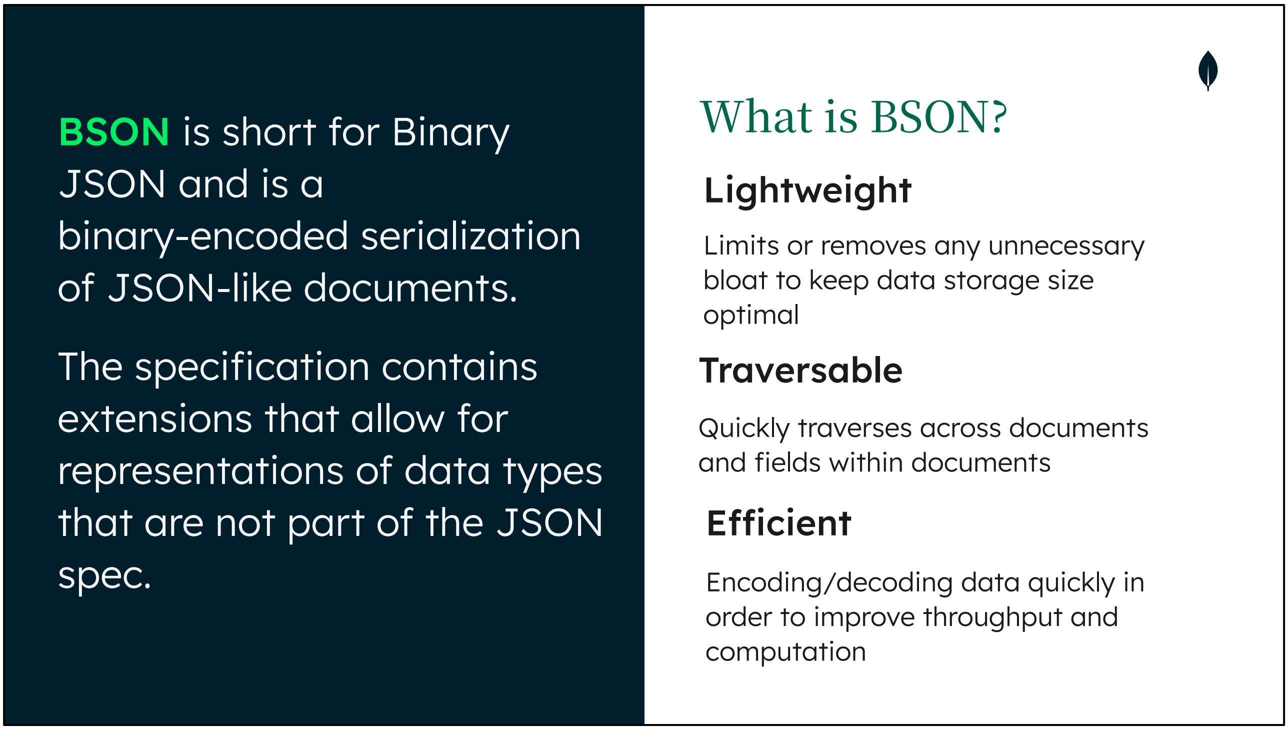
How is data stored?

A document typically stores information about one object and any of its related metadata.

Documents store data in field-value pairs. The values can be a variety of types and structures, including strings, numbers, dates, arrays, or objects.

Documents can be stored in formats like BSON, JSON, and XML.





What is BSON?

BSON is short for Binary JSON and is a binary-encoded serialization of JSON-like documents.

The specification contains extensions that allow for representations of data types that are not part of the JSON spec.

Lightweight

Limits or removes any unnecessary bloat to keep data storage size optimal

Traversable

Quickly traverses across documents and fields within documents

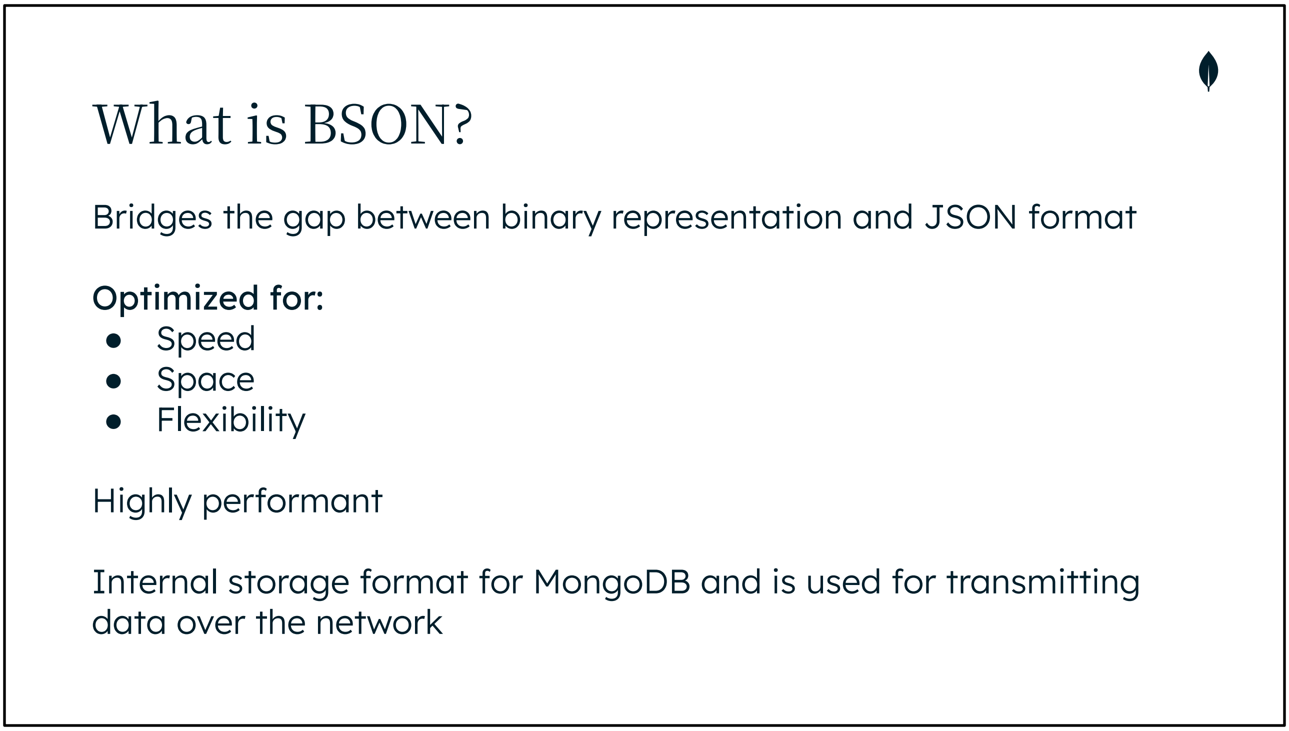
Efficient

Encoding/decoding data quickly in order to improve throughput and computation

JSON is just how we tend to present documents, but it actually has very little to do with how they are stored/processed within MongoDB.

BSON, or binary JSON, is actually how data is stored within the database and it is also how data is transmitted across the network.

It is a very lightweight format which keeps the smallest data size possible, it was designed to be easily traversable, and as a binary format it is highly computational and efficient in terms of encoding/decoding.



What is BSON?

Bridges the gap between binary representation and JSON format

Optimized for:

- Speed
- Space
- Flexibility

Highly performant

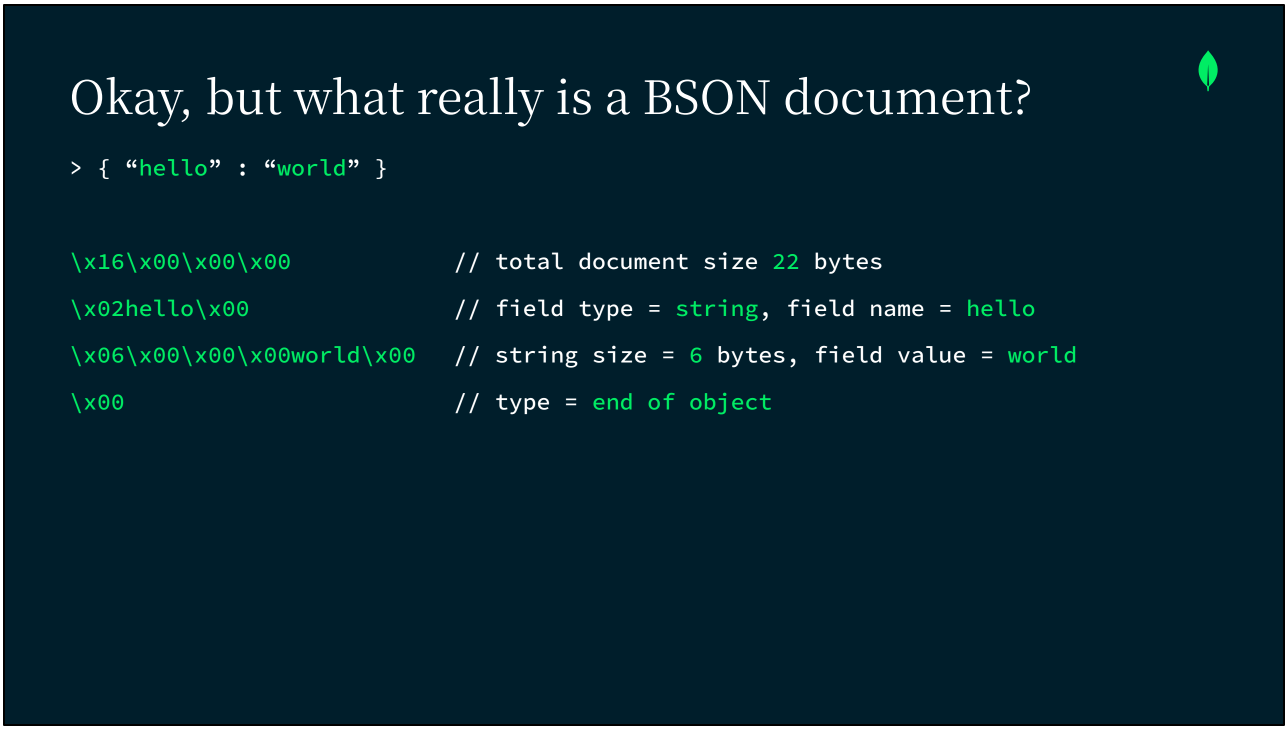
Internal storage format for MongoDB and is used for transmitting data over the network

JSON is human readable but space and speed inefficient. Binary JSON, or BSON, was developed to address these shortcomings.

BSON is optimized for speed and space to facilitate both efficient storage but also transmission across the network.

BSON is highly performant due to the design to facilitate the traversal of data which enables fast retrieval as well.

The key point to note is that BSON is the underlying storage format that data is written to using MongoDB.



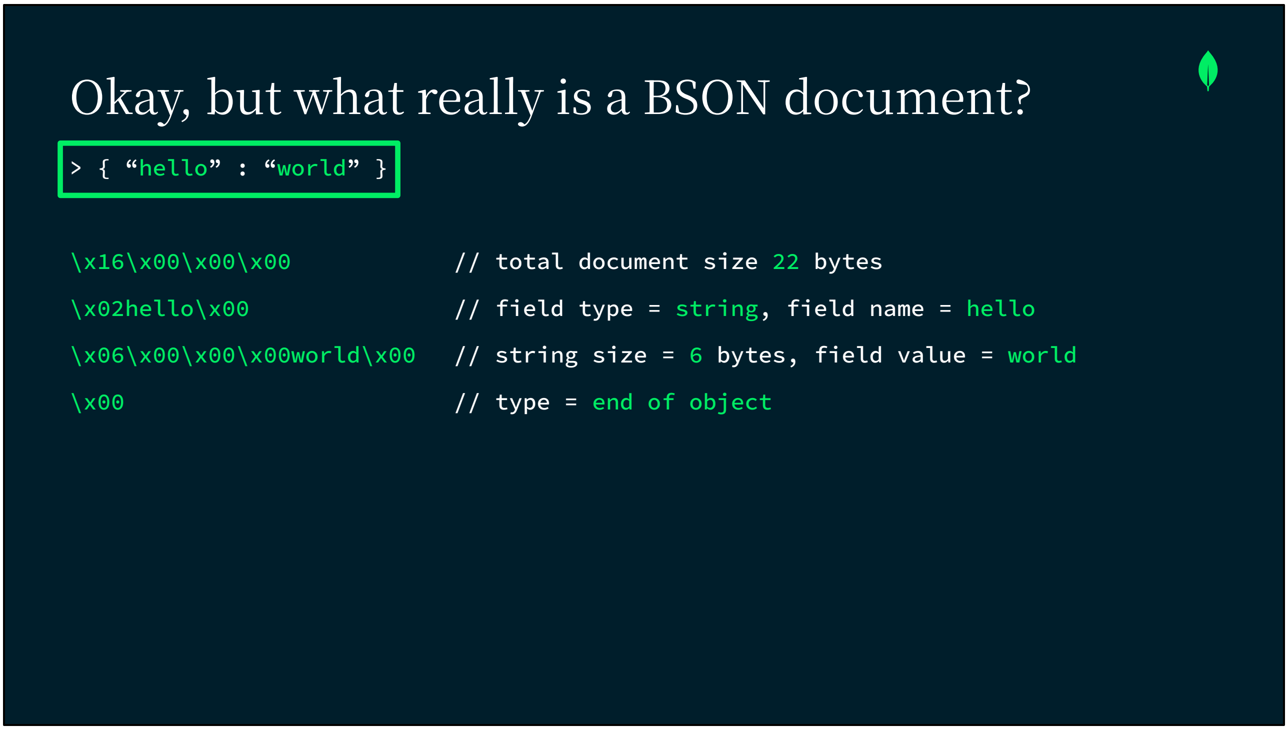
Okay, but what really is a BSON document?

```
> { “hello” : “world” }

\x16\x00\x00\x00             // total document size 22 bytes
\x02hello\x00                // field type = string, field name = hello
\x06\x00\x00\x00world\x00    // string size = 6 bytes, field value = world
\x00                         // type = end of object
```

Let’s look at the traditional “Hello World” example, firstly we have it represented in JSON and then as encoded in BSON below.

See: http://bsonspec.org/faq.html



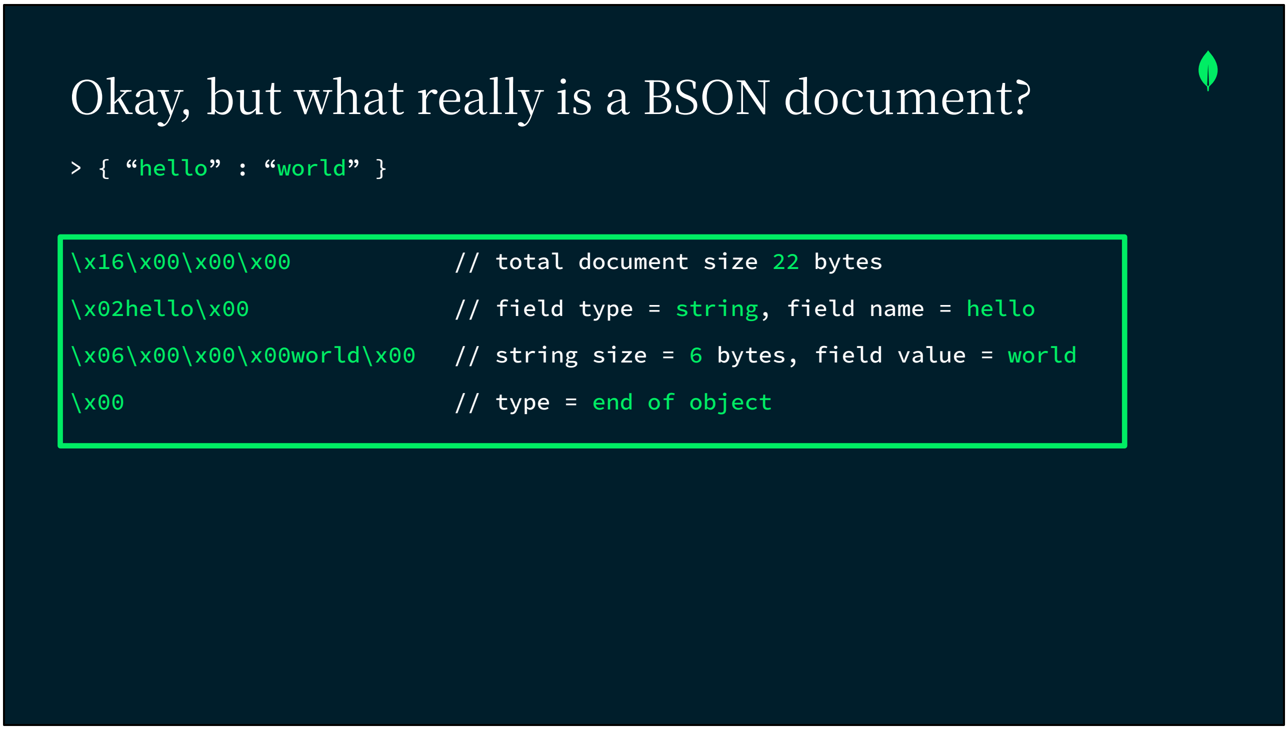
Okay, but what really is a BSON document?

```
> { “hello” : “world” }
```

```
\x16\x00\x00\x00             // total document size 22 bytes
\x02hello\x00                // field type = string, field name = hello
\x06\x00\x00\x00world\x00    // string size = 6 bytes, field value = world
\x00                         // type = end of object
```

Here’s a example of the JSON {“hello”:”world”} highlighted by the green box.

See: http://bsonspec.org/faq.html



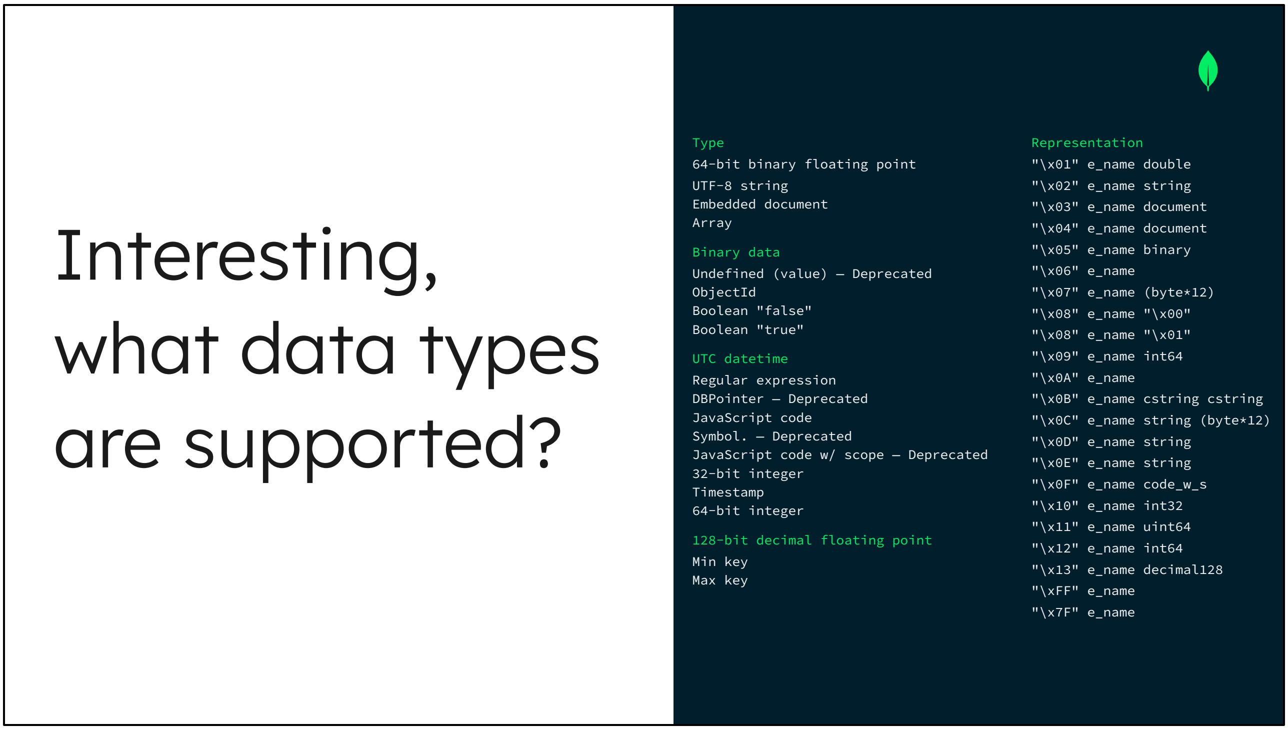
Okay, but what really is a BSON document?

```
> { “hello” : “world” }
```

```
\x16\x00\x00\x00             // total document size 22 bytes
\x02hello\x00                // field type = string, field name = hello
\x06\x00\x00\x00world\x00    // string size = 6 bytes, field value = world
\x00                         // type = end of object
```

Here is the BSON representation of “Hello World” when it is encoded. The document size is the first item, then the type of the field (a string), then the field name (hello), and the length of the field string. This is followed by the field value and then the indicator that it is the end of the object.

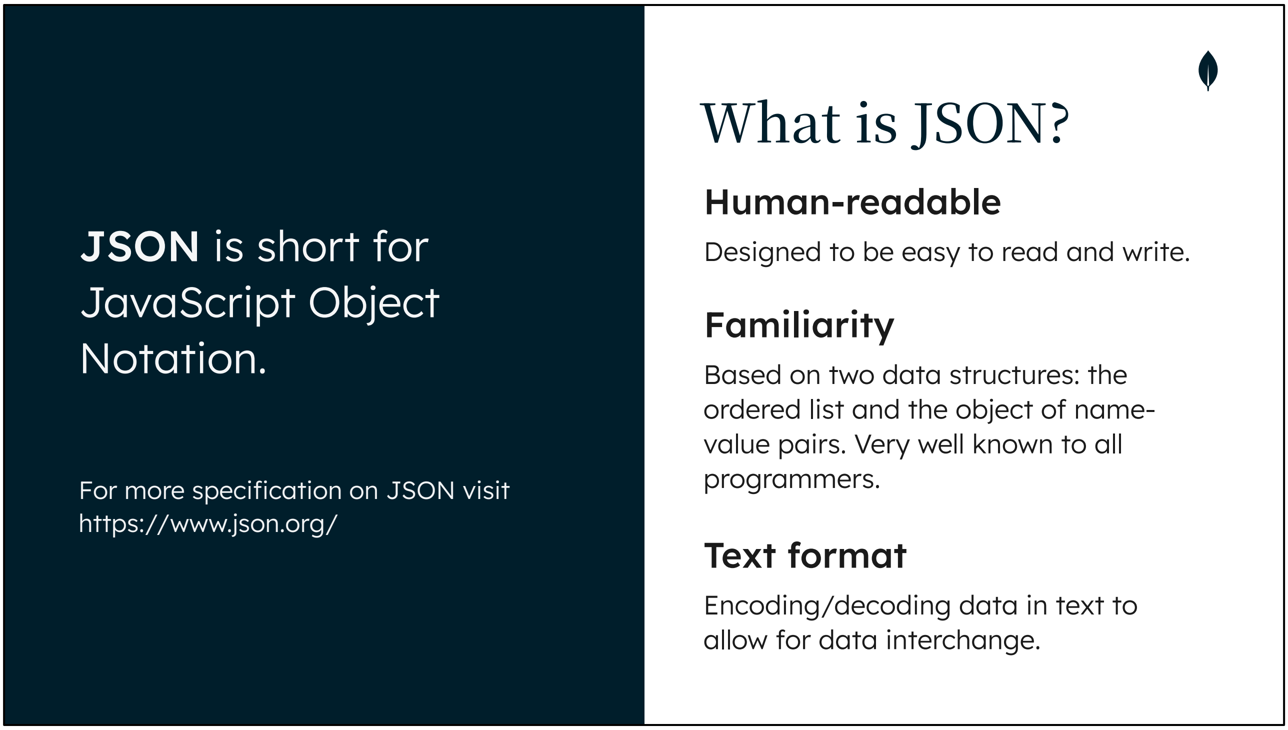
See: http://bsonspec.org/faq.html



Interesting, what data types are supported?

Type	Representation
64-bit binary floating point	"\x01" e_name double
UTF-8 string	"\x02" e_name string
Embedded document	"\x03" e_name document
Array	"\x04" e_name document
Binary data	"\x05" e_name binary
Undefined (value) – Deprecated	"\x06" e_name
ObjectId	"\x07" e_name (byte*12)
Boolean "false"	"\x08" e_name "\x00"
Boolean "true"	"\x08" e_name "\x01"
UTC datetime	"\x09" e_name int64
Regular expression	"\x0A" e_name
DBPointer – Deprecated	"\x0B" e_name cstring cstring
JavaScript code	"\x0C" e_name string (byte*12)
Symbol. – Deprecated	"\x0D" e_name string
JavaScript code w/ scope – Deprecated	"\x0E" e_name string
32-bit integer	"\x0F" e_name code_w_s
Timestamp	"\x10" e_name int32
64-bit integer	"\x11" e_name uint64
128-bit decimal floating point	"\x12" e_name int64
Min key	"\x13" e_name decimal128
Max key	"\xFF" e_name
	"\x7F" e_name

BSON offers a number of additional data types beyond JSON, examples include the decimal128 for floating point decimal.



JSON is short for JavaScript Object Notation.

For more specification on JSON visit https://www.json.org/

What is JSON?

Human-readable

Designed to be easy to read and write.

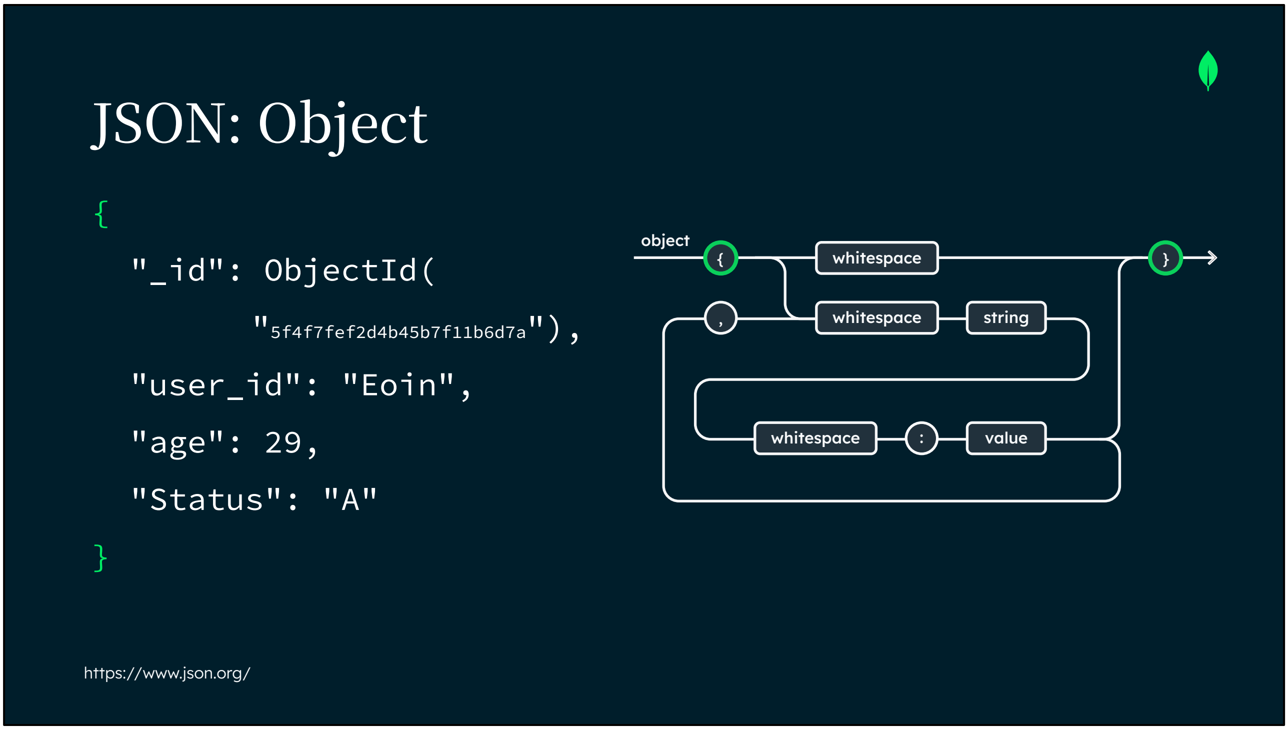
Familiarity

Based on two data structures: the ordered list and the object of name-value pairs. Very well known to all programmers.

Text format

Encoding/decoding data in text to allow for data interchange.

JSON, uses text encoding, making it human readable, but is slower to parse computations when compared to BSON. Based on two data structures, the ordered list and the object of name-value pairs. JSON is very well known to all programmers.
The syntax of JSON is very well known and intuitive to most developers, it follows similar patterns to other popular programming languages.
It uses curly brackets to mark the start and the end of the document.
MongoDB refers to keys as fields. The field-value pairs in a document are separated by colons (:).
Each field must be enclosed within quotation marks. String values are often quoted as good practice.
Each field-value pair is separated within the document by commas.



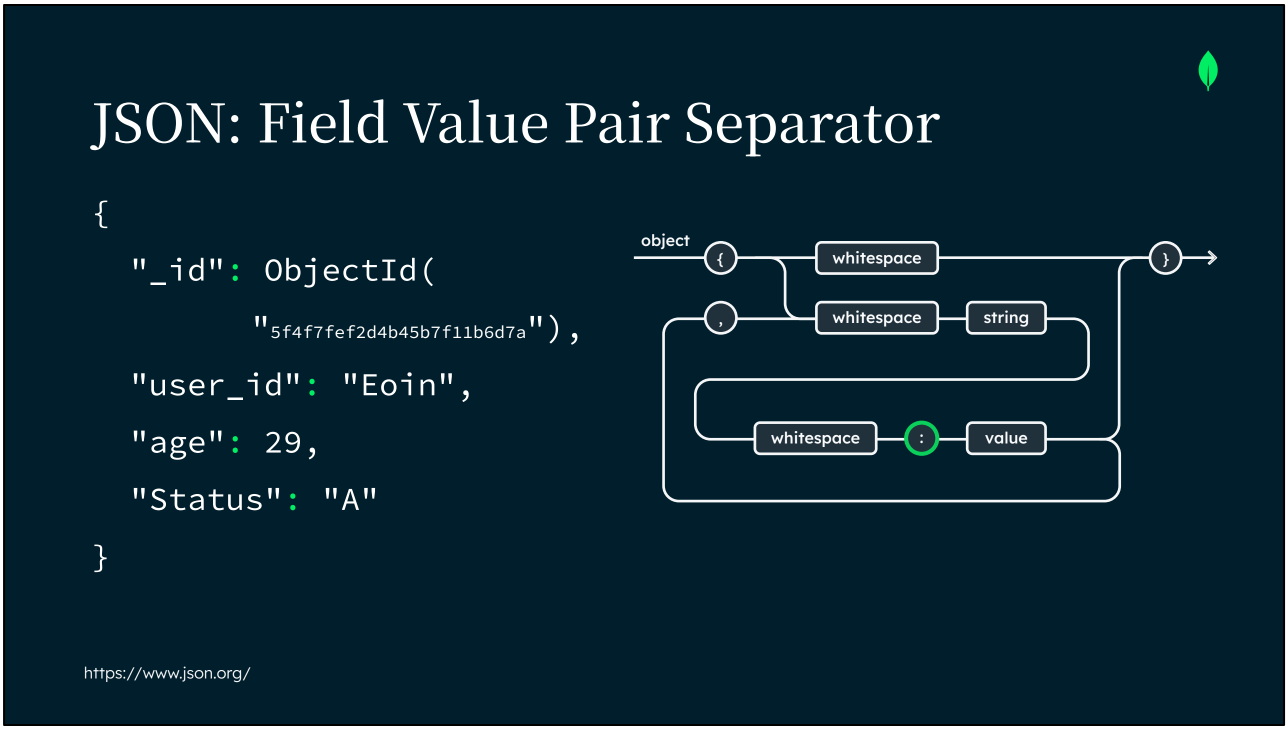
JSON: Object

```
{
  "_id": ObjectId(
         "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Eoin",
  "age": 29,
  "Status": "A"
}
```

https://www.json.org/

The JSON specification website has a number of very helpful syntax/railway diagrams that we will explore over the next few slides to better understand JSON.

The first concept to understand is that of the object which is the main container and we can see how to interpret the syntax diagram by applying the start and the end brackets from our example JSON document.

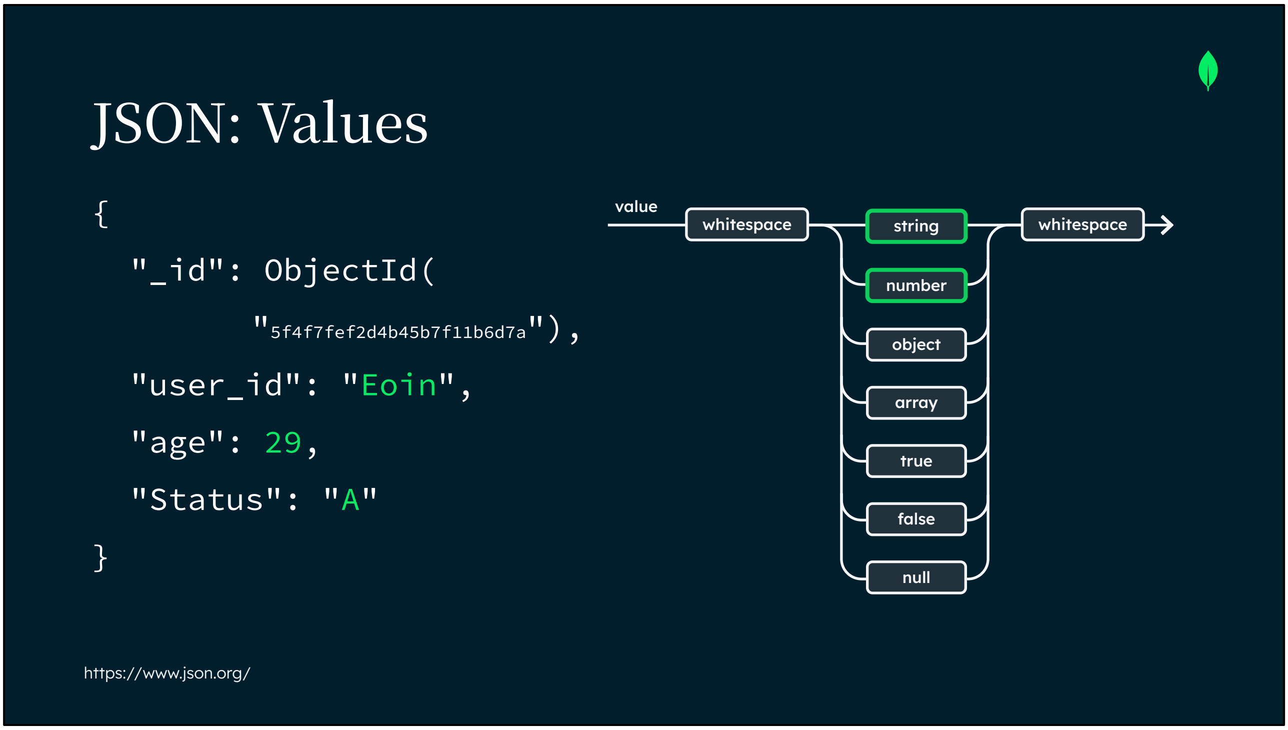


JSON: Field Value Pair Separator

```
{
  "_id": ObjectId(
         "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Eoin",
  "age": 29,
  "Status": "A"
}
```

https://www.json.org/

We can see that the syntax diagram also clearly highlights that *all* field value pairs must be separated by colons. We can again reference our example document and apply it to the syntax diagram to verify this.

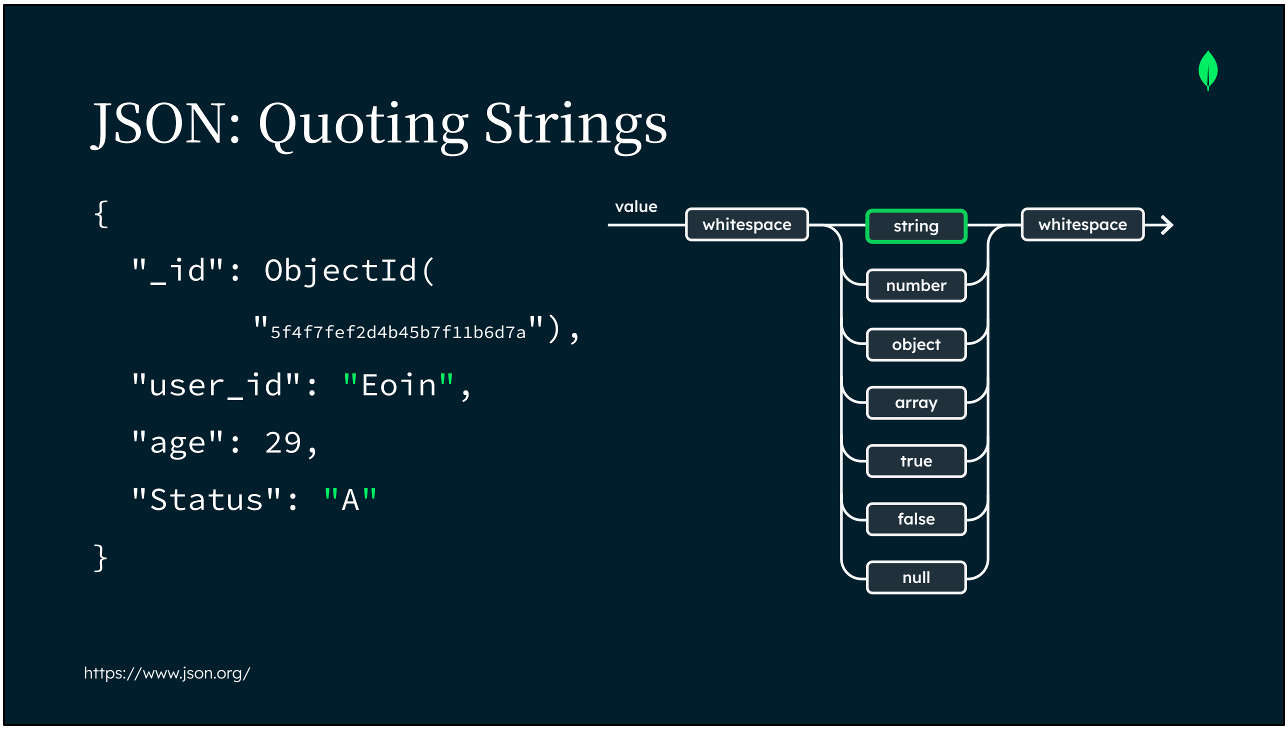


JSON: Values

```
{
  "_id": ObjectId(
         "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Eoin",
  "age": 29,
  "Status": "A"
}
```

https://www.json.org/

The value syntax diagram for JSON highlights the range of possible types for the value. We highlight the string and the number types that correspond to examples highlighted in our sample document.

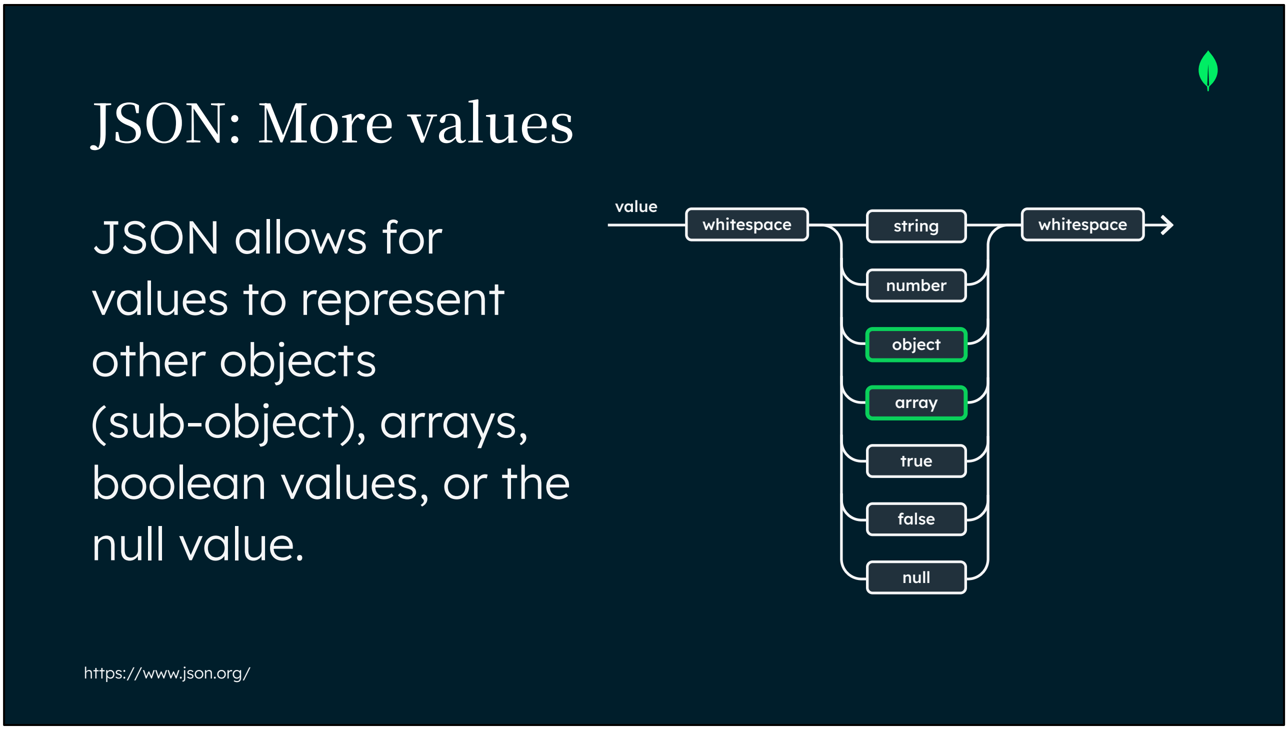


JSON: Quoting Strings

```
{
  "_id": ObjectId(
          "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Eoin",
  "age": 29,
  "Status": "A"
}
```

https://www.json.org/

An good practice for string values in JSON is to quote the values within them to allow for multi-word space separated text. This allows a sentence, or indeed paragraphs, to be stored in the string value rather than just a single word or letter.

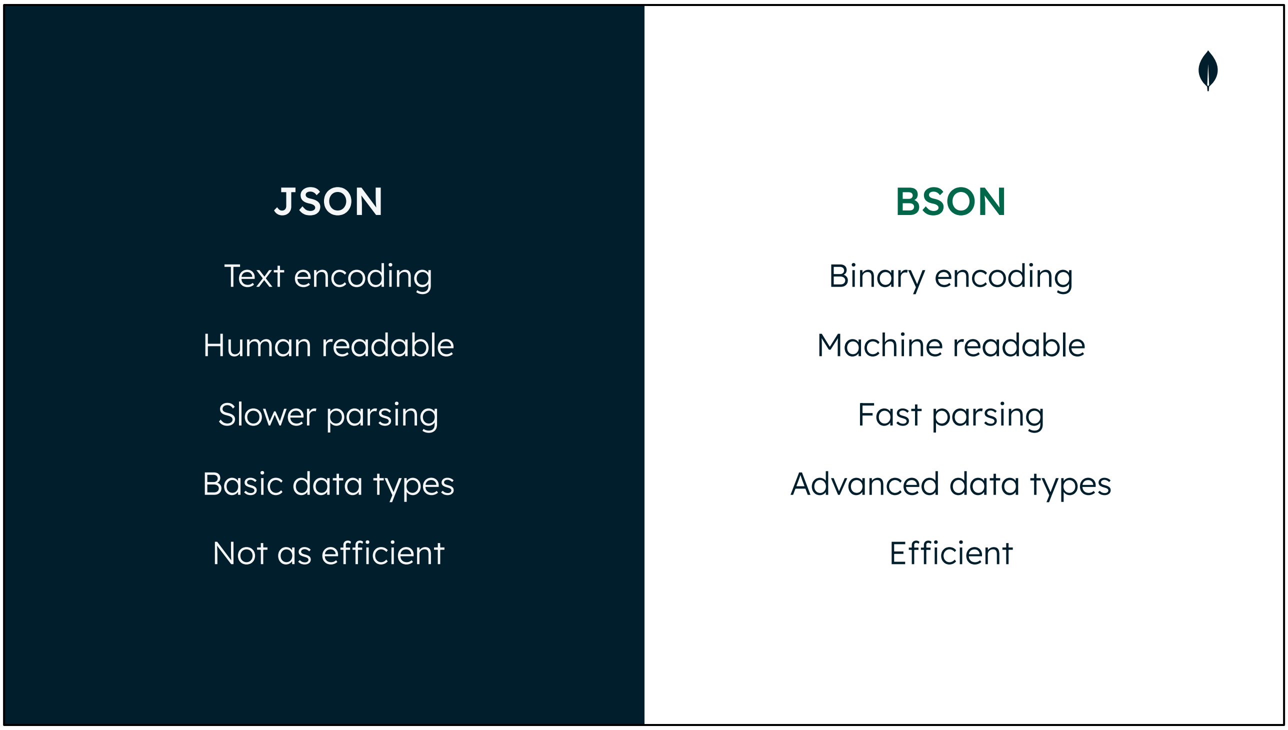


JSON: More values

JSON allows for values to represent other objects (sub-object), arrays, boolean values, or the null value.

https://www.json.org/

In the syntax diagram we further see that we could have a sub-object (sub-document), an array, the boolean true or the boolean false, as well as null as valid values for the value side of the field value (key value) pair.



JSON	BSON
Text encoding	Binary encoding
Human readable	Machine readable
Slower parsing	Fast parsing
Basic data types	Advanced data types
Not as efficient	Efficient

Let’s compare JSON and BSON broadly. The efficiency and fast parsing aspects of BSON are key in understanding why it is the storage format used. It is clear as to why JSON is used as the presentation format in the MongoSh, MongoDB Compass, or Atlas’s Data Explorer.

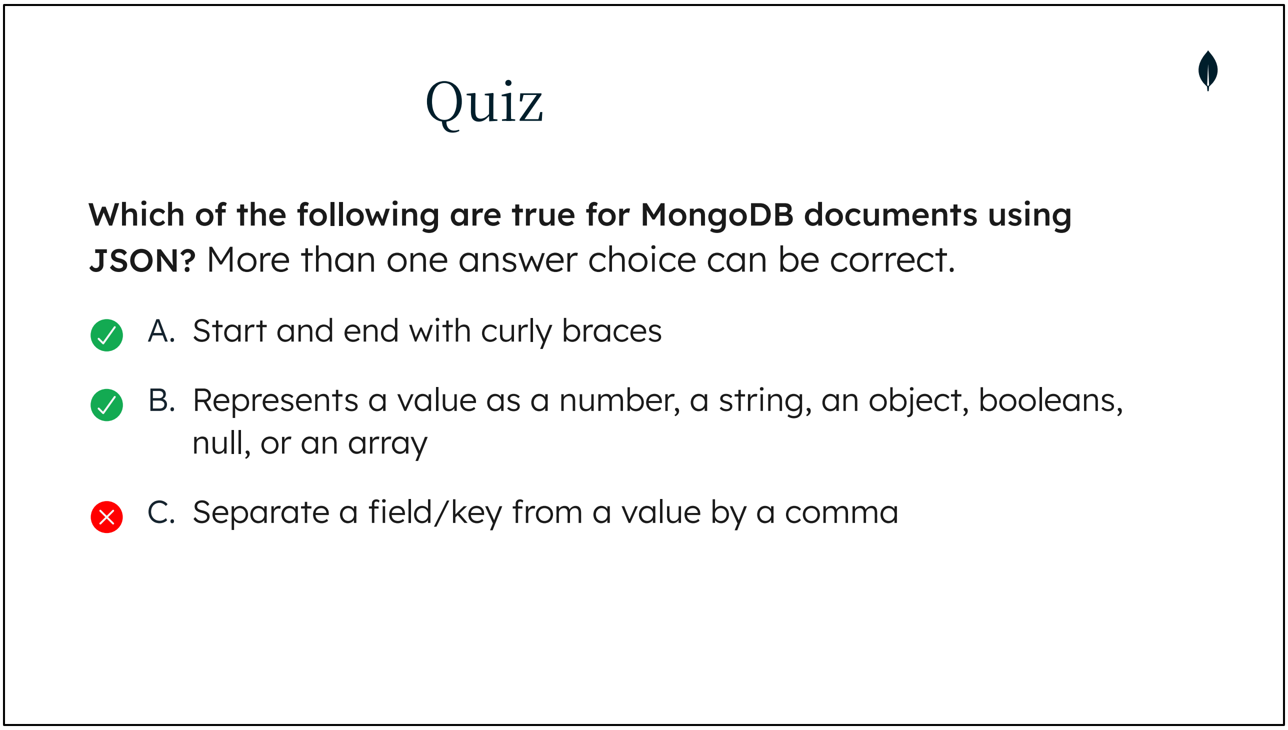
JSON uses text encoding, whilst BSON is binary encoded.
JSON is human readable, whilst BSON is not and is only machine readable.
JSON is slower to parse computational when compared to BSON.
BSON has a richer set of data types than JSON has to represent data.
JSON is just not as efficient as BSON as JSON isn’t a binary format which is more efficient in terms of computational processing.

Quiz

Quiz

Which of the following are true for MongoDB documents using JSON? More than one answer choice can be correct.

- ● A. Start and end with curly braces
- ● B. Represents a value as a number, a string, an object, booleans, null, or an array
- ● C. Separate a field/key from a value by a comma



Quiz

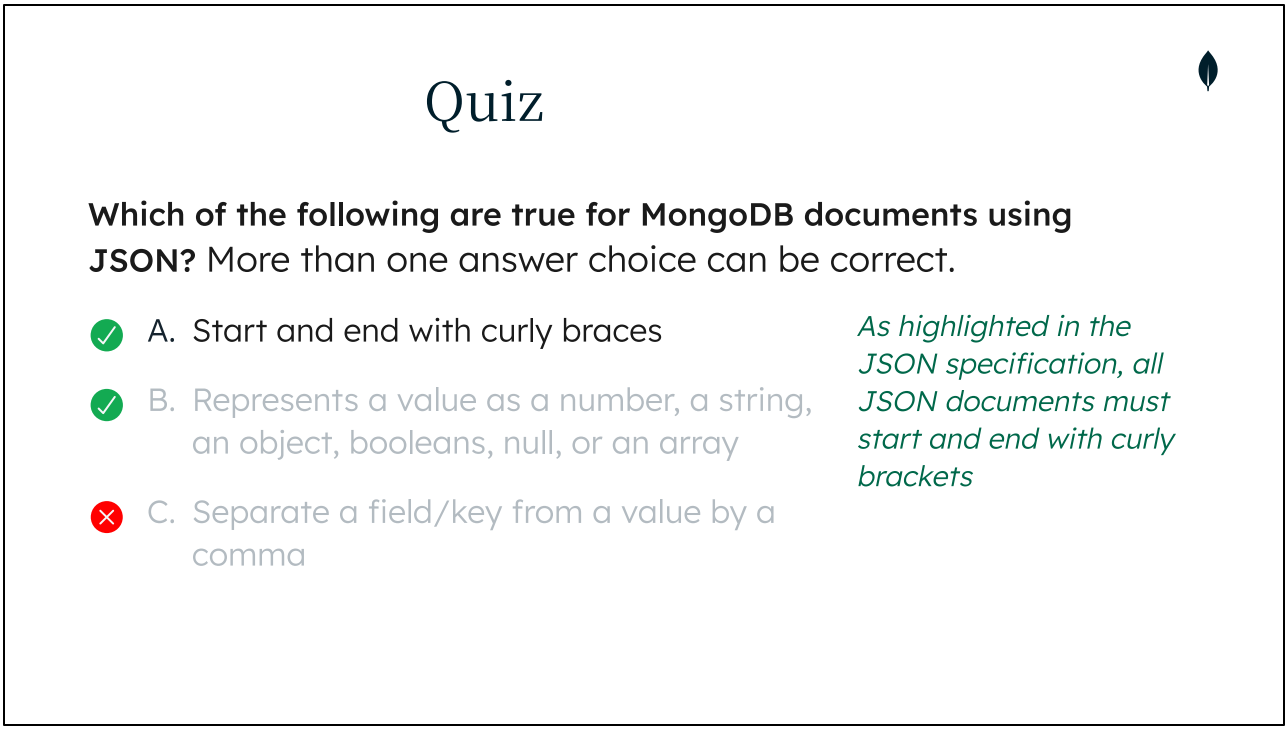
Which of the following are true for MongoDB documents using JSON? More than one answer choice can be correct.

✓ A. Start and end with curly braces

✓ B. Represents a value as a number, a string, an object, booleans, null, or an array

✗ C. Separate a field/key from a value by a comma

CORRECT: Start and end with curly braces - As highlighted in the JSON specification, all JSON documents must start and end with curly brackets.
CORRECT: Represents a value as a number, a string, an object, booleans, null, or an array - A JSON document value can hold any of these data types per the JSON specification.
INCORRECT: Separate a field/key from a value by a comma - A field/key is separated by a colon ":" rather than a comma "," from the corresponding value. Commas are used to separate field-value pairs.



Quiz

Which of the following are true for MongoDB documents using JSON? More than one answer choice can be correct.

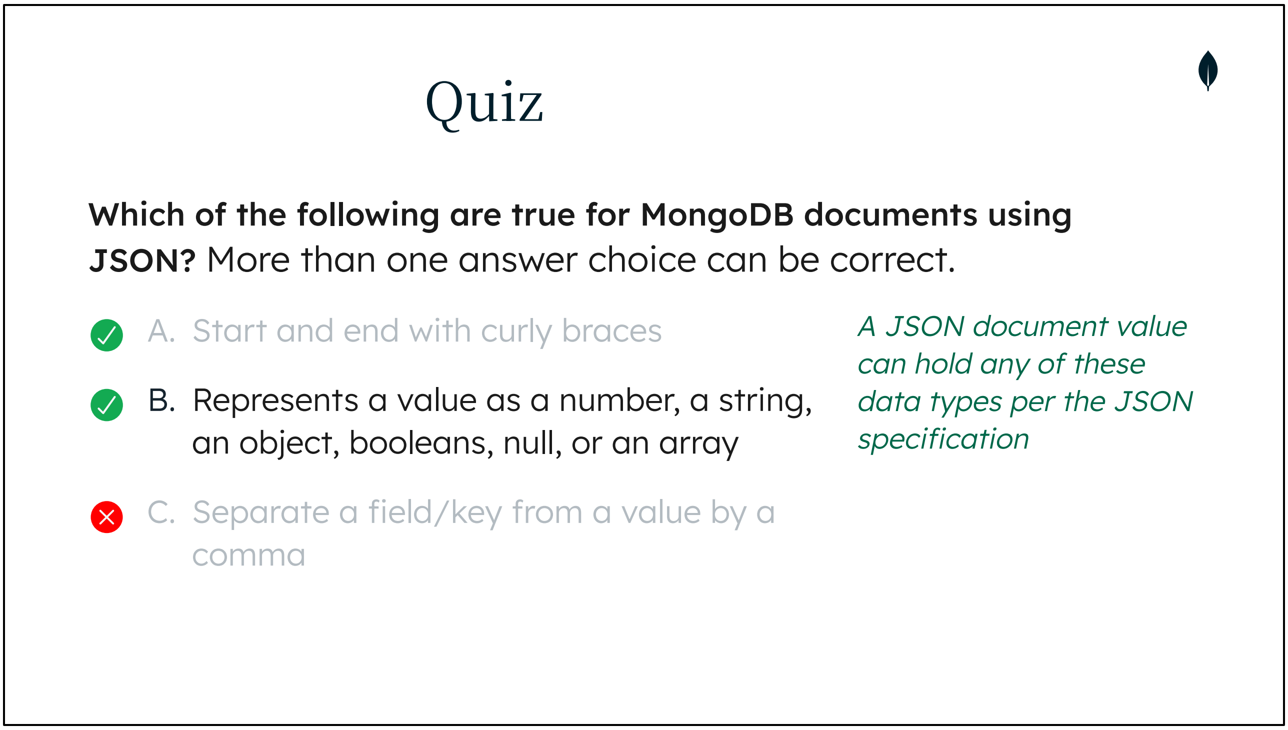
✓ A. Start and end with curly braces

✓ B. Represents a value as a number, a string, an object, booleans, null, or an array

✗ C. Separate a field/key from a value by a comma

As highlighted in the JSON specification, all JSON documents must start and end with curly brackets

CORRECT: Start and end with curly braces - As highlighted in the JSON specification, all JSON documents must start and end with curly brackets.



Quiz

Which of the following are true for MongoDB documents using JSON? More than one answer choice can be correct.

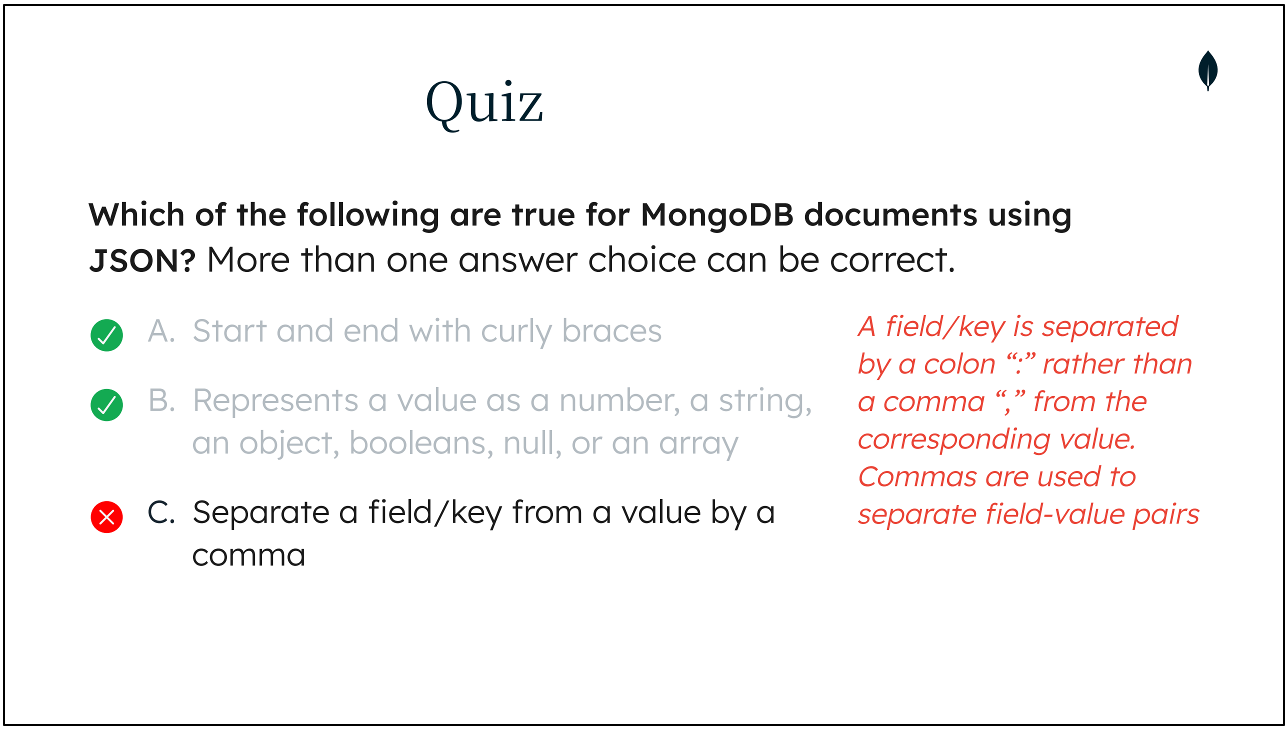
✓ A. Start and end with curly braces

✓ B. Represents a value as a number, a string, an object, booleans, null, or an array

✗ C. Separate a field/key from a value by a comma

A JSON document value can hold any of these data types per the JSON specification

CORRECT: Represents a value as a number, a string, an object, booleans, null, or an array - A JSON document value can hold any of these data types per the JSON specification.



Quiz

Which of the following are true for MongoDB documents using JSON? More than one answer choice can be correct.

✓ A. Start and end with curly braces

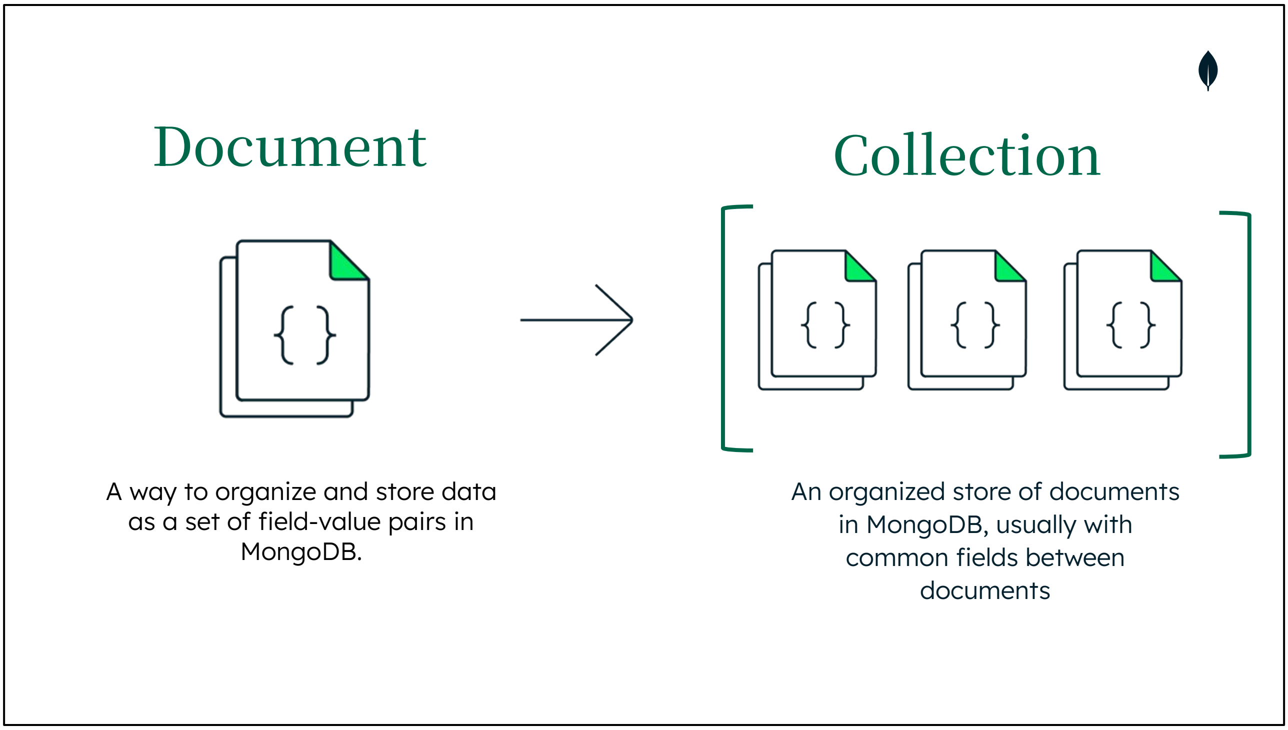
✓ B. Represents a value as a number, a string, an object, booleans, null, or an array

✗ C. Separate a field/key from a value by a comma

A field/key is separated by a colon ":" rather than a comma "," from the corresponding value. Commas are used to separate field-value pairs

INCORRECT: Separate a field/key from a value by a comma - A field/key is separated by a colon ":" rather than a comma "," from the corresponding value. Commas are used to separate field-value pairs.

Collections in the Document Model



Document

A way to organize and store data as a set of field-value pairs in MongoDB.

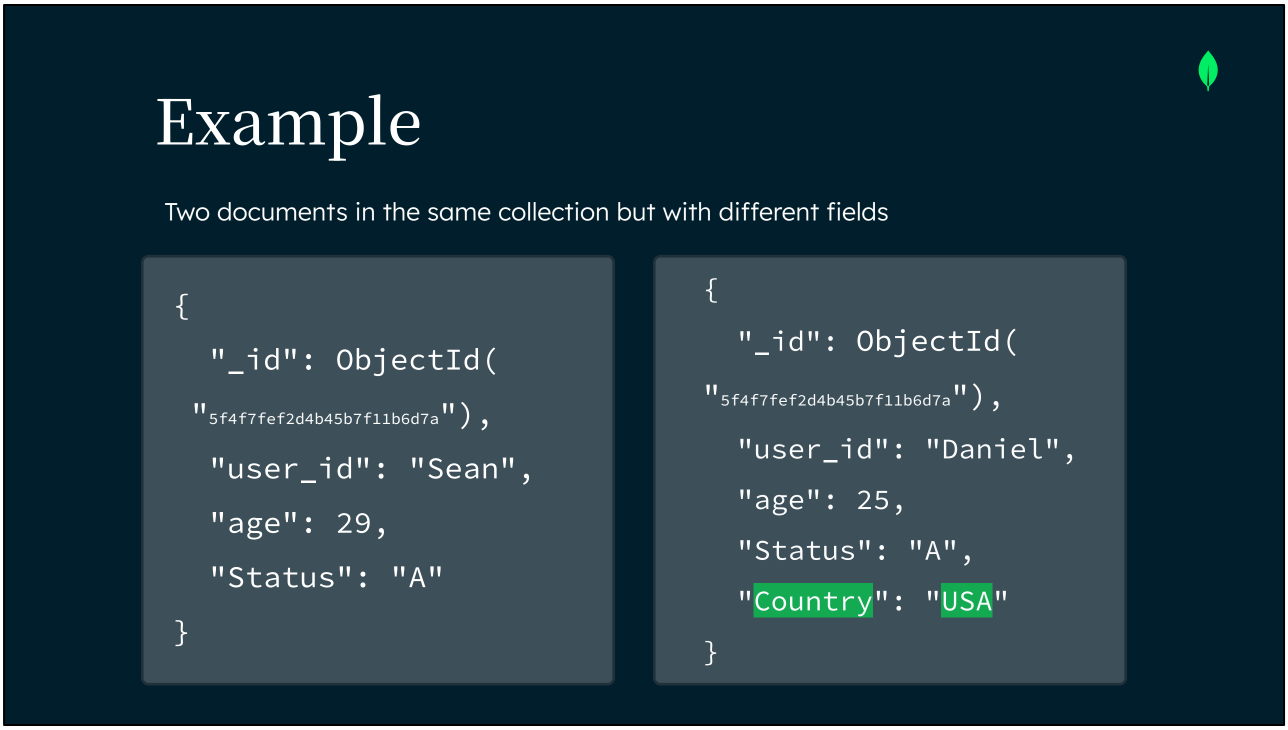
Collection

An organized store of documents in MongoDB, usually with common fields between documents

Another way data is stored in MongoDB’s document model is through collections. A collection is a group of documents. Collections typically store documents that have similar contents. In MongoDB, these usually have common fields between documents but this is not a requirement unless you are using schema validation to enforce specific common fields.

A document is a way to organize and store data as a set of field-value pairs in MongoDB.

A collection is an organized store of documents in MongoDB, these usually have common fields between documents but this is not a requirement unless you are using schema validation to enforce specific common fields.



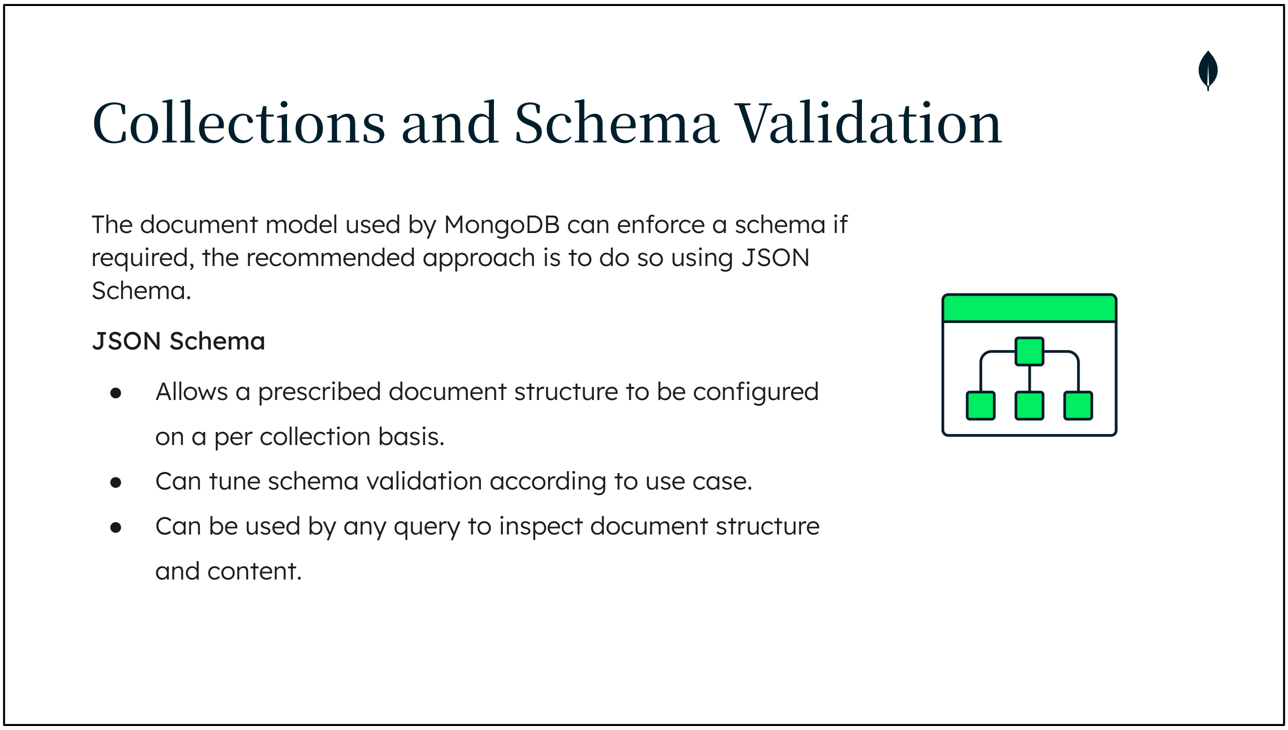
Example

Two documents in the same collection but with different fields

```
{
  "_id": ObjectId(
 "5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Sean",
  "age": 29,
  "Status": "A"
}
```

```
{
  "_id": ObjectId(
"5f4f7fef2d4b45b7f11b6d7a"),
  "user_id": "Daniel",
  "age": 25,
  "Status": "A",
  "Country": "USA"
}
```

MongoDB collections do not by default enforce a single schema on a collection so whilst documents can have common fields, they are not required to have the same fields. There is no issue having two documents where the first document has a “country” field and the second document does not have the “country” field.



Collections and Schema Validation

The document model used by MongoDB can enforce a schema if required, the recommended approach is to do so using JSON Schema.

JSON Schema

- Allows a prescribed document structure to be configured on a per collection basis.
- Can tune schema validation according to use case.
- Can be used by any query to inspect document structure and content.

MongoDB can enforce a schema if required, the recommended approach is to do so using JSON Schema. This allows a prescribed document structure to be configured on a per collection basis.

Document validation allows restrictions to be made when new content is added, it allows for the presence, the type, and the values to be validated as part of this process as well.

Schema validation in MongoDB has tunable controls. Administrators have the flexibility to tune schema validation according to use case – for example, if a document fails to comply with the defined structure, it can either be rejected, or still written to the collection while logging a warning message. Structure can be imposed on just a subset of fields – for example, requiring a valid customer name and address, while others fields can be freeform, such as the social media handle and cellphone number. And, validation can be turned off entirely, allowing complete schema flexibility

The schema definition can be used by any query to inspect document structure and content. For example, DBAs can identify all documents that do not conform to a prescribed schema.

This avoids having to implement this validation logic in your application or in middleware.

Data modeling is critical to setting up any database to meet the needs of an application, but in a document based non-relational database, such as MongoDB, there is great flexibility on how to model the data. How do you know which way to store your data?

We will cover some best practices when it comes to modeling data in MongoDB.

Quiz

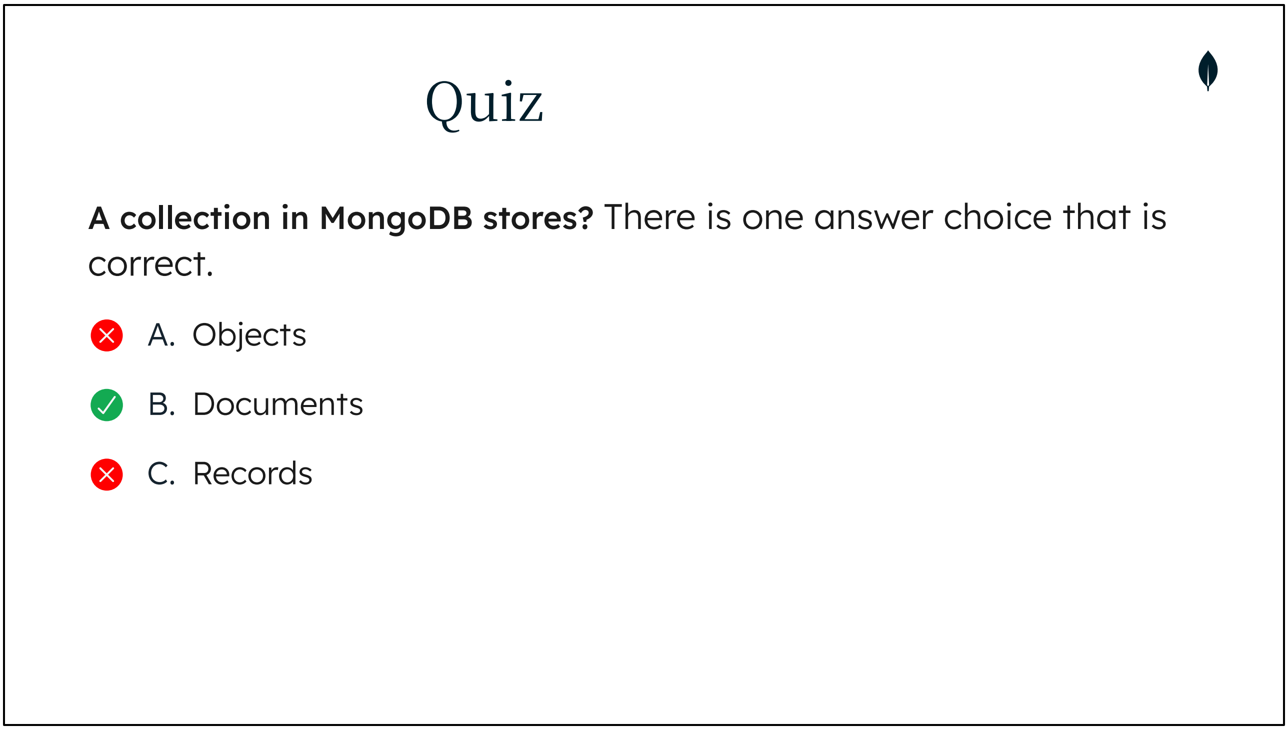
Quiz

A collection in MongoDB stores? There is one answer choice that is correct.

○ A. Objects

○ B. Documents

○ C. Records

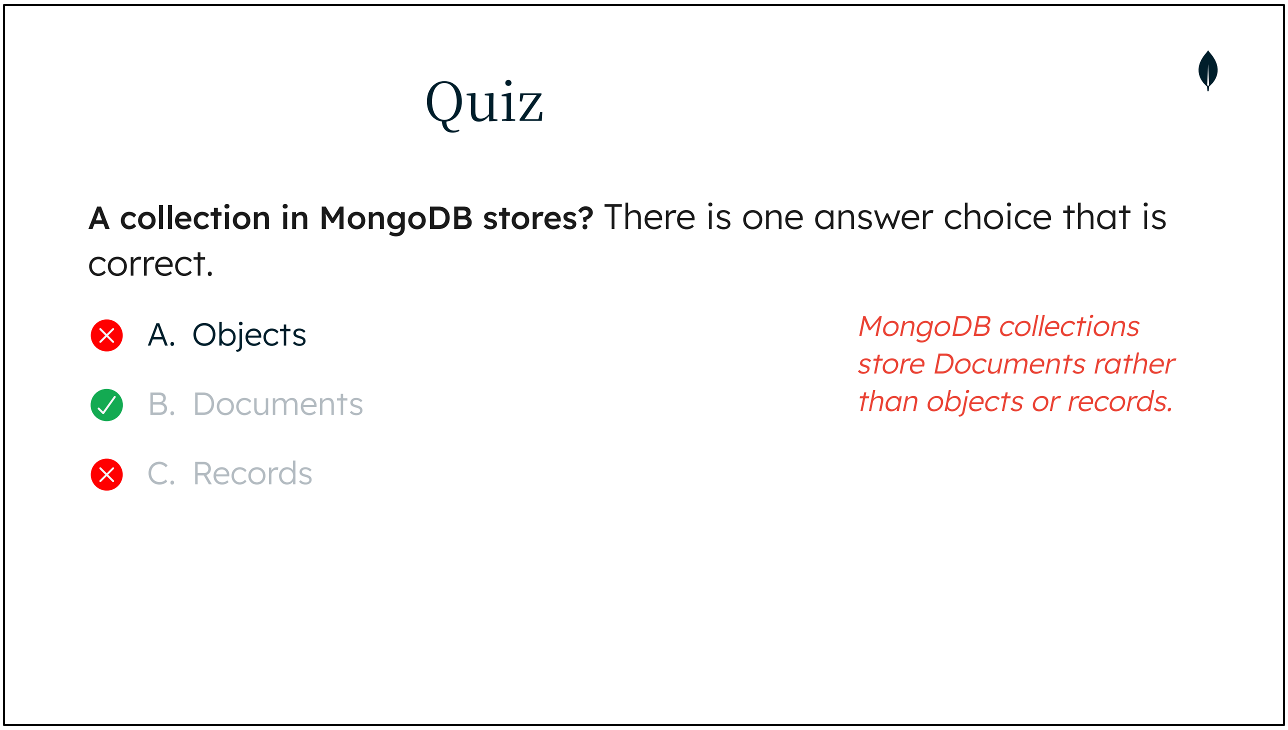


Quiz

A collection in MongoDB stores? There is one answer choice that is correct.

- ✗ A. Objects
- ✓ B. Documents
- ✗ C. Records

INCORRECT: Objects - MongoDB collections store documents rather than objects or record.
CORRECT: Documents - A MongoDB collection holds all the documents for that specific collection.
INCORRECT: Records - A relational database stores records in a table, this is not how MongoDB collections store data.



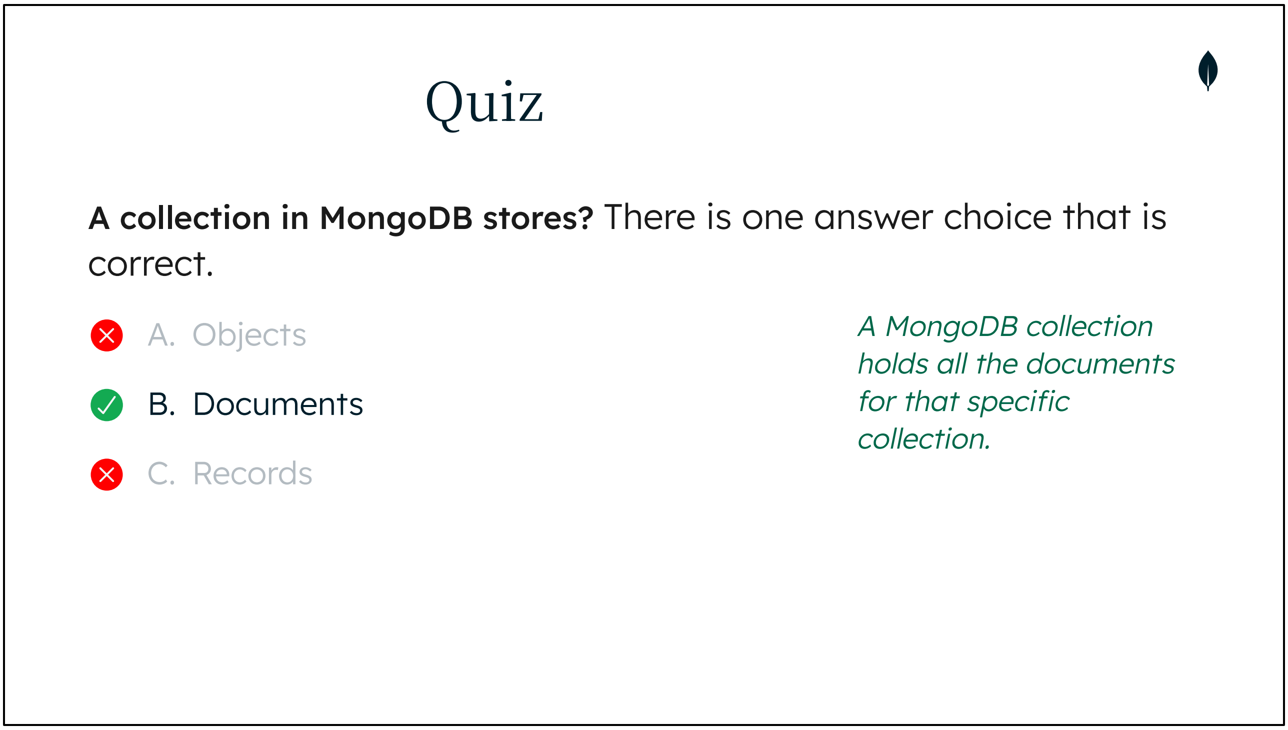
Quiz

A collection in MongoDB stores? There is one answer choice that is correct.

- ✗ A. Objects
- ✓ B. Documents
- ✗ C. Records

MongoDB collections store Documents rather than objects or records.

INCORRECT: Objects - MongoDB collections store Documents rather than objects or records.



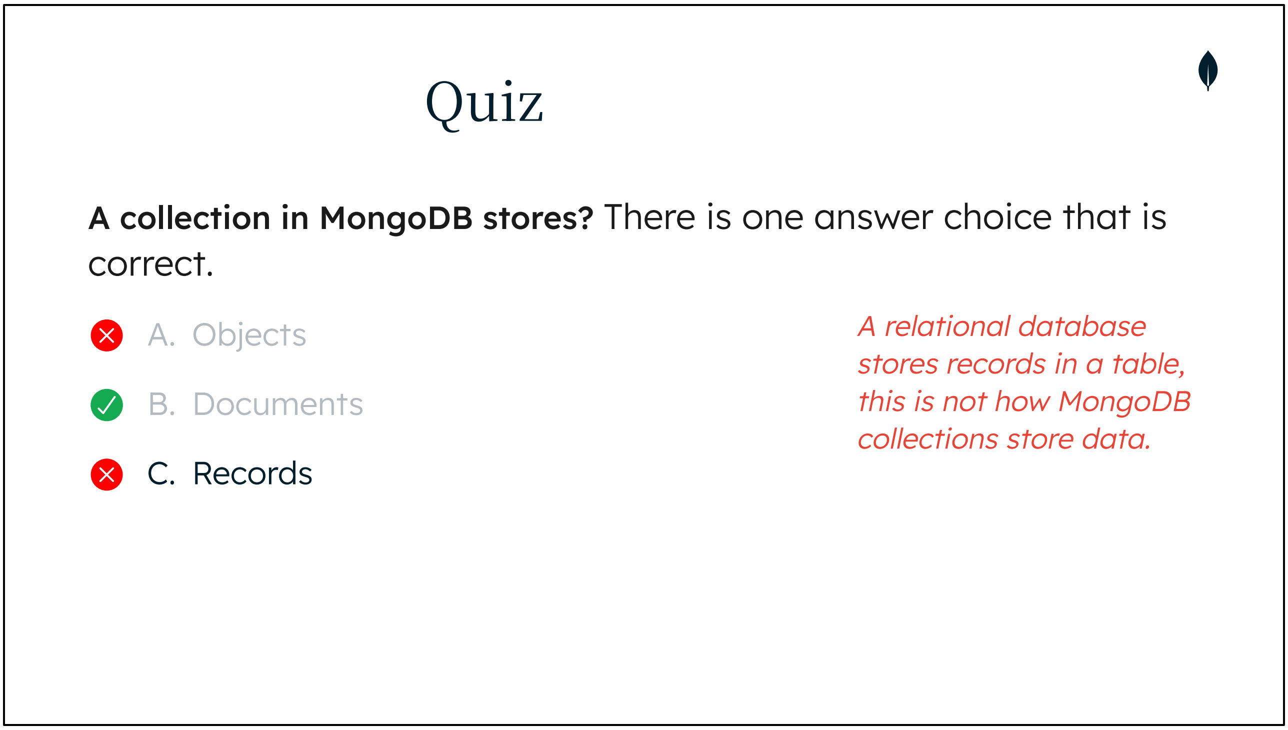
Quiz

A collection in MongoDB stores? There is one answer choice that is correct.

- ✗ A. Objects
- ✓ B. Documents
- ✗ C. Records

A MongoDB collection holds all the documents for that specific collection.

CORRECT: Documents - A MongoDB collection holds all the documents for that specific collection.



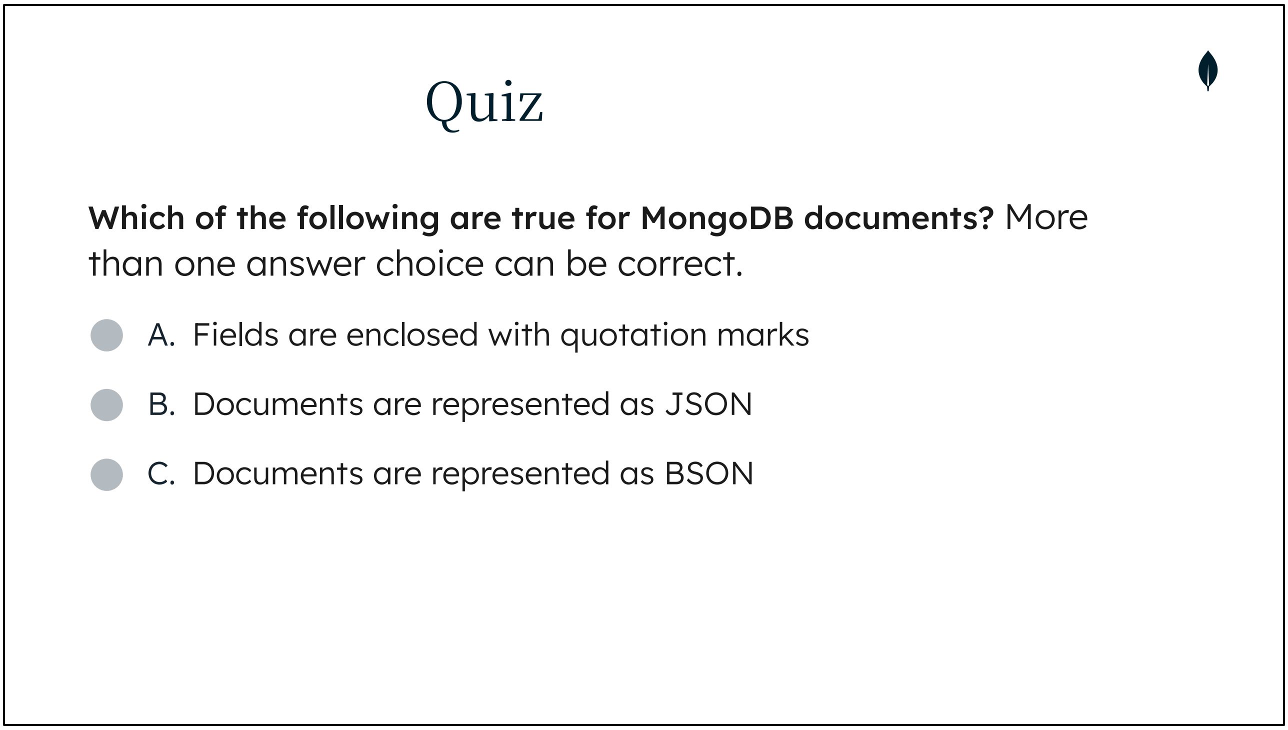
Quiz

A collection in MongoDB stores? There is one answer choice that is correct.

- ✗ A. Objects
- ✓ B. Documents
- ✗ C. Records

A relational database stores records in a table, this is not how MongoDB collections store data.

INCORRECT: Records - A relational database stores records in a table, this is not how MongoDB collections stores data.



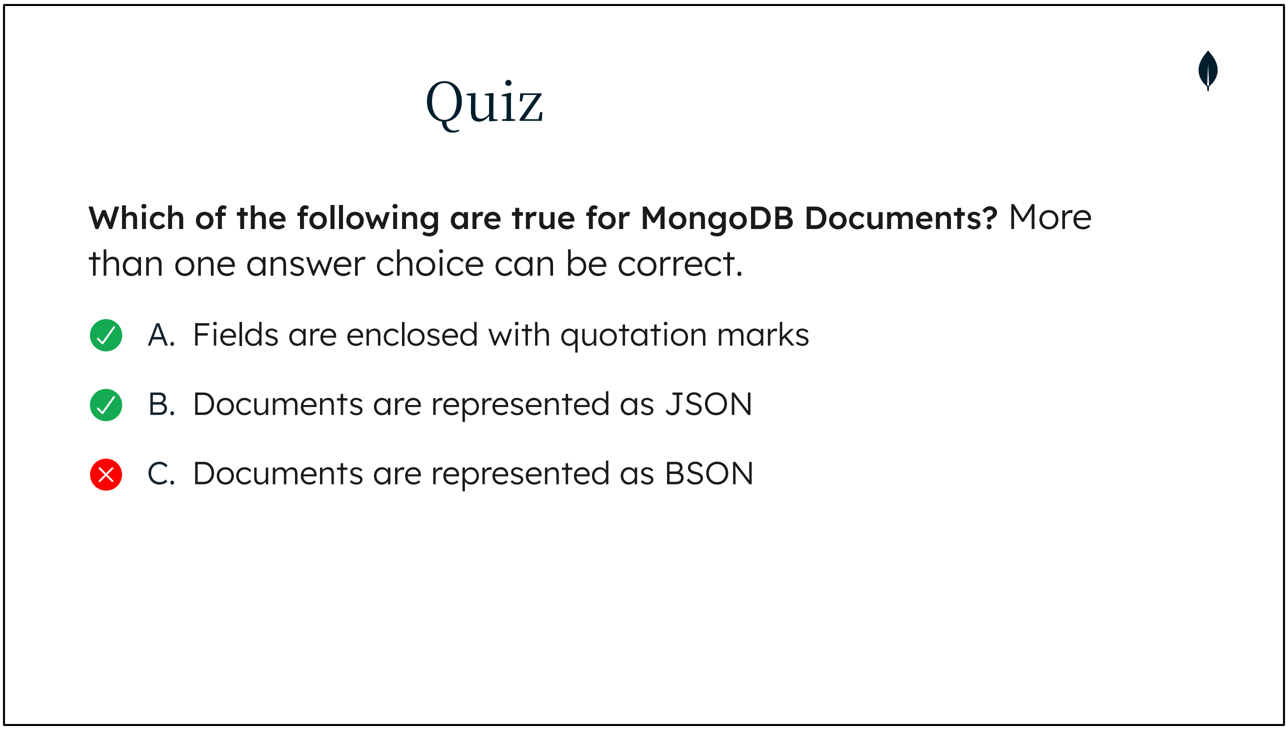
Quiz

Which of the following are true for MongoDB documents? More than one answer choice can be correct.

● A. Fields are enclosed with quotation marks

● B. Documents are represented as JSON

● C. Documents are represented as BSON



Quiz

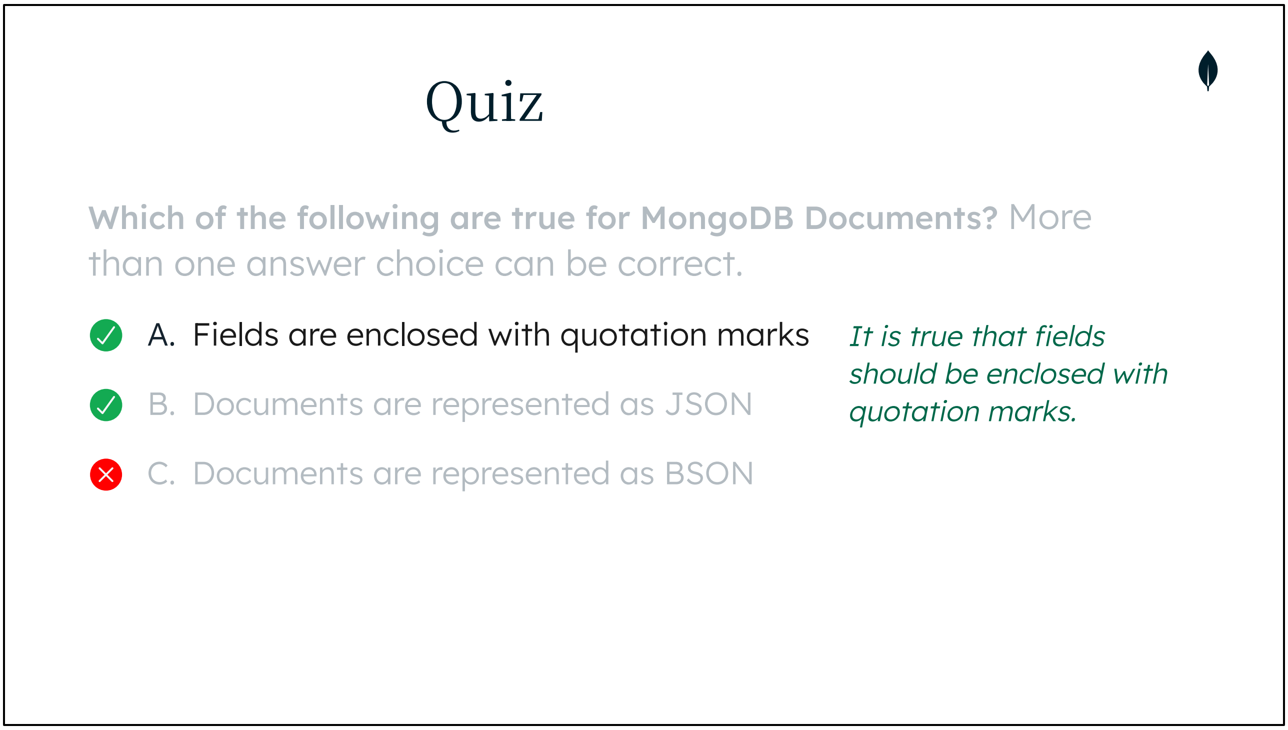
Which of the following are true for MongoDB Documents? More than one answer choice can be correct.

✓ A. Fields are enclosed with quotation marks

✓ B. Documents are represented as JSON

✗ C. Documents are represented as BSON

CORRECT: Field are enclosed with quotation marks - each field must be enclosed with quotation marks per the JSON specification, which we'll cover immediately after this quiz.
CORRECT: Documents are represented as JSON - Documents are represented in MongoDB as JSON to simplify their readability, however they are stored as BSON. We'll explore JSON and then BSON in the next lessons.
INCORRECT: Documents are represented as BSON - Documents are represented as JSON however they are **stored as BSON**.



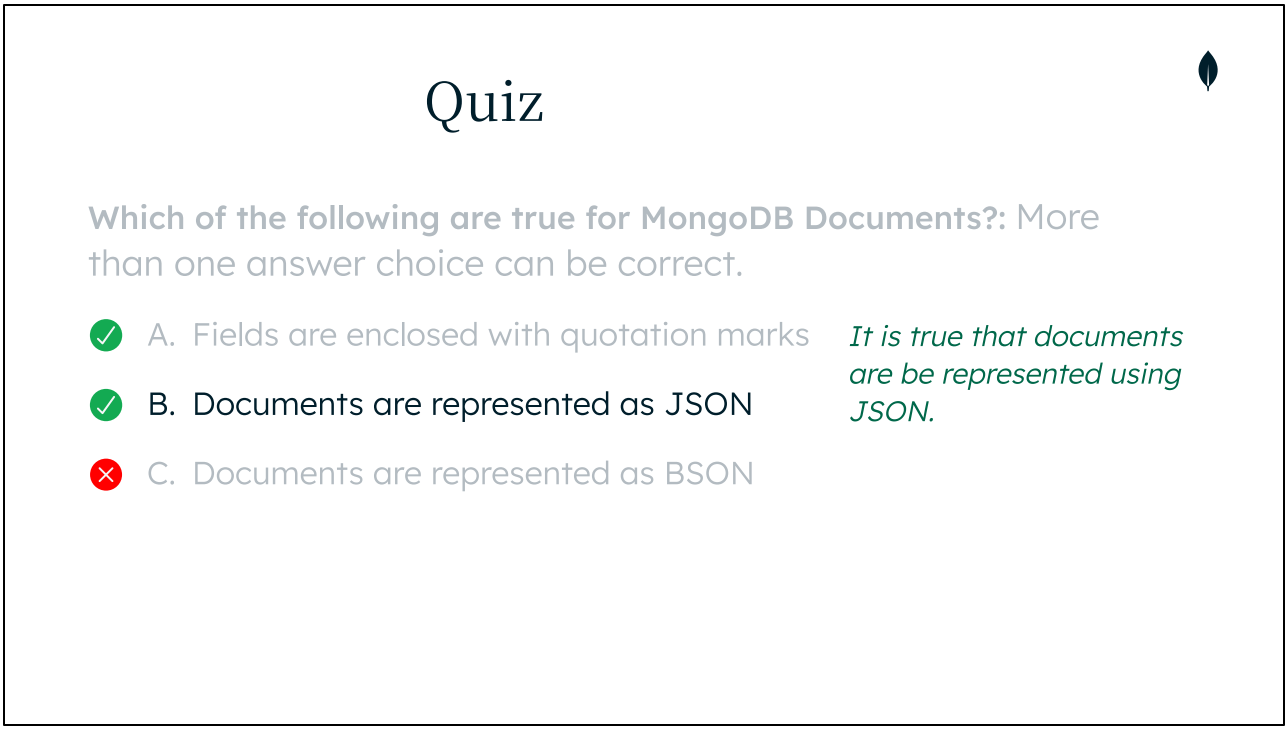
Quiz

Which of the following are true for MongoDB Documents? More than one answer choice can be correct.

- ✓ A. Fields are enclosed with quotation marks
- ✓ B. Documents are represented as JSON
- ✗ C. Documents are represented as BSON

It is true that fields should be enclosed with quotation marks.

CORRECT: Field are enclosed with quotation marks - This is correct. Each field must be enclosed with quotation marks per the JSON specification, which we’ll cover immediately after this quiz.



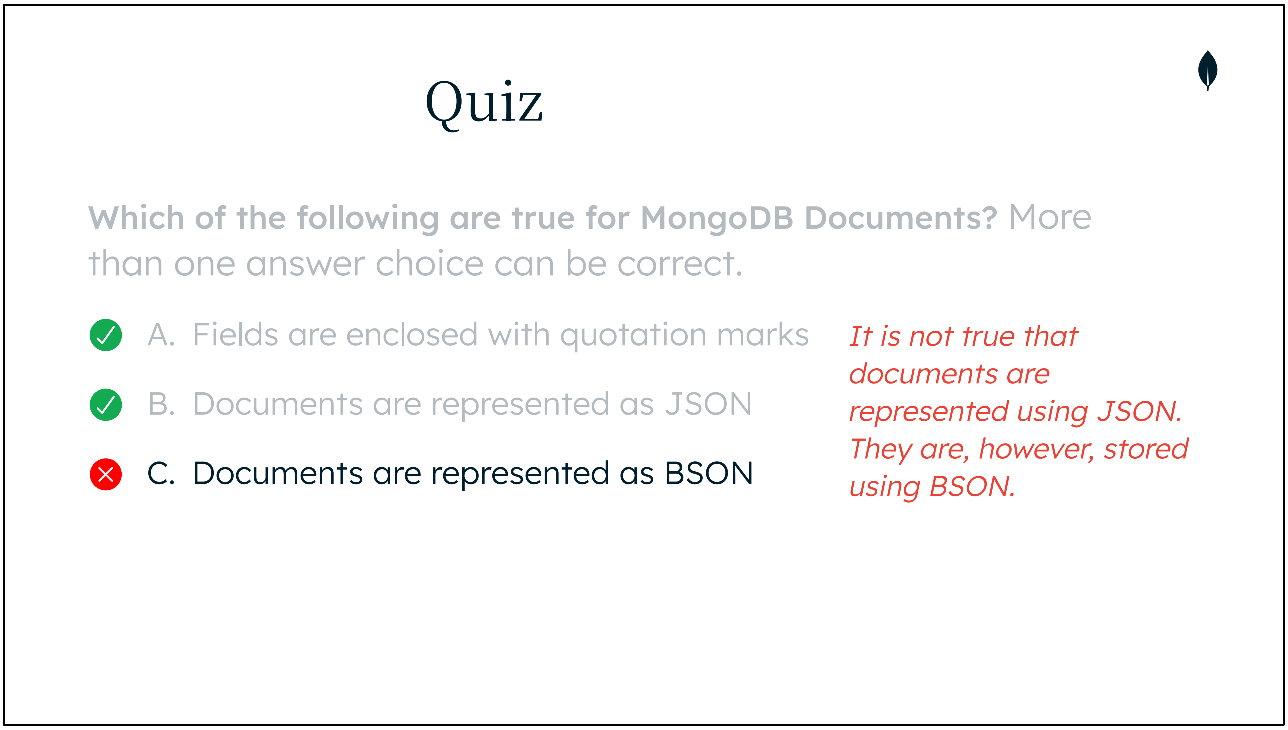
Quiz

Which of the following are true for MongoDB Documents?: More than one answer choice can be correct.

- ✓ A. Fields are enclosed with quotation marks
- ✓ B. Documents are represented as JSON
- ✗ C. Documents are represented as BSON

It is true that documents are be represented using JSON.

CORRECT: Documents are represented as JSON - Documents are represented in MongoDB as JSON to simplify their readability, however they are stored as BSON. We'll explore JSON and then BSON in the next lessons.



Quiz

Which of the following are true for MongoDB Documents? More than one answer choice can be correct.

- ✓ A. Fields are enclosed with quotation marks
- ✓ B. Documents are represented as JSON
- ✗ C. Documents are represented as BSON

It is not true that documents are represented using JSON. They are, however, stored using BSON.

INCORRECT: Documents are represented as BSON - Documents are represented as JSON however they are **stored as BSON**.

Continue Learning!

MongoDB University has free self-paced courses and labs ranging from beginner to advanced levels.

GitHub Student Developer Pack

Sign up for the MongoDB Student Pack to receive $50 in Atlas credits and free certification!

This concludes the material for this lesson. However, there are many more ways to learn about MongoDB and non-relational databases, and they are all free! Check out MongoDB's University page to find free courses that go into more depth about everything MongoDB and non-relational. For students and educators alike, MongoDB for Academia is here to offer support in many forms. Check out our educator resources and join the Educator Community. Students can receive $50 in Atlas credits and free certification through the GitHub Student Developer Pack.