MongoDB Fundamentos

Módulo 3

JavaScript en MongoDB

Utilizar JavaScript en MongoDB Shell

El shell de MongoDB es una **interfaz interactiva de Javascript.** Como tal, brinda la capacidad de usar código JavaScript directamente en el shell o ejecutarlo como un archivo independiente. En los temas subsiguientes - que tratan sobre el uso del shell para acceder a la base de datos y crear y manipular colecciones y documentos - se proporcionan ejemplos que están escritos en JavaScript. Para seguirlos, es necesario **comprender al menos algunos de los aspectos fundamentales del lenguaje.**

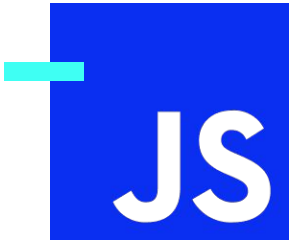
Se verán algunos de los conceptos básicos, como **variables, funciones y objetos.** Este recorrido no pretende ser una guía completa de JavaScript, sino una sinopsis de la sintaxis y los modismos importantes. Será de gran utilidad para los alumnos que no estén familiarizados con el lenguaje, ya que obtendrán información para comprender los mecanismos de consulta e interacción hacia la base de datos, desde un script con JavaScript.

Variables

Definición de variables

Para comenzar con JavaScript, lo primero es definir variables.

Las variables son un **medio para nombrar los datos,** almacenarlos y acceder temporalmente a ellos. Las variables pueden apuntar a **tipos de datos simples, como números o cadenas, o a otros más complejos, como objetos.**



Sintaxis de la definición de una variable: se utiliza la palabra clave **var** y luego se escribe el nombre que se le dará, como en este ejemplo:

```
var myData;
```

También puede **asignar un valor a la variable en la misma línea.** Por ejemplo, el siguiente código crea una variable denominada `myString` y le asigna el valor de "`Some Text`":

```
var myString = "Un texto";
```

El código anterior funciona tan bien como éste:

```
var myString;
myString = "Un texto";
```

Después de haber declarado la variable, **se puede usar su nombre para asignarle un valor y para acceder a ese valor.** Por ejemplo, el siguiente código almacena una cadena en la variable `myString` y luego la usa al asignar el valor a la variable `newString`:

```
var myString = "Un texto";
var newString = myString + "Más texto";
```

Los nombres de variables deben describir los datos almacenados en ellos para que luego pueda sea fácil utilizarlos en su programa. Las únicas reglas para crear nombres de variables son:

Deben comenzar con una letra, $, o _ y no pueden contener espacios. Importante: los nombres de las variables **distinguen entre mayúsculas y minúsculas**, por lo que `myString` es diferente de `MyString`.

Comprender los tipos de datos de JavaScript

JavaScript usa **tipos de datos** para determinar cómo manejar los datos que se asignan a una variable. El tipo de variable determina qué operaciones se pueden realizar con ella, como bucles o ejecución.

A continuación veremos los tipos más comunes de variables.

String

Esta variable almacena **cadenas de caracteres.** Los datos de caracteres se especifican mediante comillas simples o dobles. Todos los datos contenidos en las comillas se asignan a la variable de cadena.

Por ejemplo:

```
var myString = 'Un texto';
var anotherString = 'Más texto';
```

Number

Los datos se almacenan como **valor numérico.** Los números son útiles en conteos, cálculos y comparaciones.

Algunos ejemplos:

```
var myInteger = 1;
var cost = 1.33;
```

Boolean

Esta variable almacena **un solo bit que es `true` o `false`.** Los booleanos se utilizan a menudo para las banderas. Por ejemplo, se puede establecer una variable como `false` al comienzo de algún código y luego verificarla al finalizar para ver si la ejecución llegó a un punto determinado. A continuación vemos un ejemplo de definición de una variable `true` y una variable `false`:

```
var yes = true;
var no = false;
```

Array

Un `array` o matriz indexada es **una serie de elementos de datos distintos,** almacenados bajo un solo nombre de variable. Se puede acceder a cada elemento de la matriz mediante su índice de base cero: `array[index]`. El siguiente es un ejemplo de cómo crear una matriz simple y luego acceder al primer elemento, que se encuentra en el índice `0`.

```
var arr = ["uno", "dos", "tres"];
var first = arr[0];
```

Objeto literal

JavaScript admite la capacidad de crear y utilizar objetos literales. Cuando usa un objeto literal, puede acceder a valores y funciones en el objeto usando la sintaxis `object.property`.

El siguiente ejemplo muestra cómo crear y acceder a propiedades con un objeto literal:

```
var obj = {"nombre":"Carlos", "ocupacion":"Médico", "edad", "Desconocida"};
var nombre = obj.nombre;
```

Nulo

A veces no hay un valor para almacenar en una variable porque no se ha creado o ya no la está usando. En este momento, puede establecer la variable en `null`. **Usar `null` es mejor que asignar a la variable un valor de `0` o una cadena vacía `""`** porque esos pueden ser valores válidos para
la variable.

Asignar la variable `null` le permite no asignar ninaún valor v verificar `null` dentro de su código.

```
var newVar = null;
```

Salida de datos en un script de shell de MongoDB

Se utilizan cuatro formas para generar datos desde el script de shell de MongoDB:

```
print(data, ...);

printjson(object);

print(tojson(object));

print(JSON.stringify(object));
```

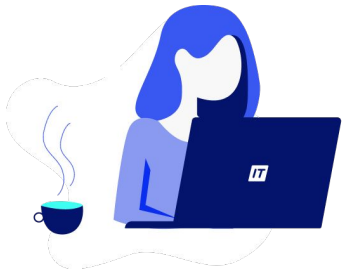
mongoDB

El método **`print()`** simplemente imprime los datos que se le pasan como argumento. Por ejemplo, la declaración:

```
print("Hola desde Mongo");
```

da como resultado esta salida:

```
Hola desde Mongo
```



Si se proporcionan varios elementos de datos, se imprimen juntos. Por ejemplo, la declaración:

```
print("Hola", "desde", "Mongo");
```

también genera esto:

```
Hola desde Mongo
```

El método **`printjson()`** imprime una bonita forma del objeto JavaScript.

Los métodos **`print(JSON.stringify(object))`** y **`print(tojson(object))`** también imprimen una forma de cadena muy compacta de un objeto JavaScript.

Operadores

Uso de operadores

Los operadores de JavaScript **permiten modificar el valor de una variable.** Los valores se asignan mediante el operador = .

JavaScript proporciona varios operadores diferentes que se pueden agrupar en **dos tipos, aritméticos y de asignación.**

Operadores aritméticos

Se utilizan para realizar operaciones entre valores variables y directos. Esta tabla enumera las operaciones aritméticas, junto con los resultados que se aplican.

Operador	Descripción	Ejemplo	Resultado en x
+	Suma	x=y+5 x=y+"5" x="Four"+y+"4"	9 "49" "Four44"
-	Resta	x=y-2	2
++	Incremento	x=y++ x=++y	4 5
- -	Decremento	x=y- - x=- - y	4 3
*	Multiplicación	x=y*4	16
/	División	x=10/y	2.5
%	Módulo (resto de la división)	x=y%3	1

Operadores aritméticos de JavaScript: resultados basados en un valor inicial y=4.

Operadores de asignación

Asignan un valor a una variable. Además del operador **=**, varios formularios permiten manipular los datos a medida que se asigna el valor. Esta tabla enumera las operaciones de asignación, junto con los resultados que se aplican.

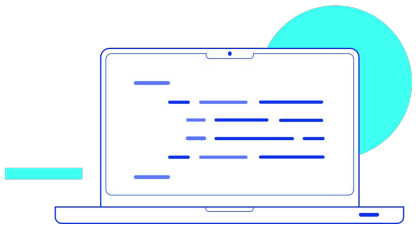
Operador	Ejemplo	Operaciones aritméticas equivalentes	Resultado en x
=	x=5	x=5	5
+=	x+=5	x=x+5	15
-=	x-=5	x=x-5	5
*=	x*=5	x=x*5	50
/=	x/=5	x=x/5	2
%=	x%=5	x=x%5	0

Operadores de asignación de JavaScript: resultados basados en el valor inicial x=10.

Aplicación de operadores condicionales y de comparación

Los condicionales ofrecen una forma de aplicar la lógica para que **cierto código** se **ejecute solo en las condicio-nes correctas.** Esto se hace aplicando la **lógica de comparación** a los valores de las variables.

Las siguientes secciones describen las comparaciones disponibles en JavaScript y explican cómo aplicarlas en declaraciones condicionales.



Operadores de comparación

Un operador de comparación evalúa dos datos y **devuelve `true` si la evaluación es correcta o `false` si la evaluación no es correcta**.

Los operadores de comparación comparan el valor a la izquierda del operador con el valor a la derecha. La forma más sencilla de comprender la sintaxis de comparación de JavaScript es ver una lista con algunos posibles casos. La siguiente tabla enumera los operadores de comparación, junto con algunos ejemplos.

Operador	Descripción	Ejemplo	Resultado
`==`	Es igual a (sólo valor)	`x==8` `x==10`	`false` `true`
`===`	Igualdad de valor y tipo	`x===10` `x==="10"`	`true` `false`
`!=`	No es igual	`x!=5`	`true`
`!==`	Desigualdad de valor y tipo	`x!=="10"` `x!==10`	`true` `false`
`>`	Es mayor a	`x>5`	`true`
`>=`	Es mayor o igual a	`x>=10`	`true`
`<`	Es menor a	`x<5`	`false`
`<=`	Es menor o igual a	`x<=10`	`true`

Operadores de comparación de JavaScript: resultados basados en un valor inicial de x=10.

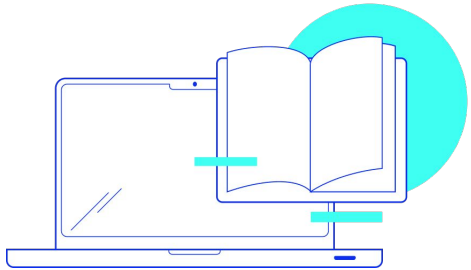
Podemos encadenar múltiples comparaciones usando operadores lógicos y paréntesis estándar. La tabla por debajo enumera los operadores lógicos y explica cómo usarlos para encadenar comparaciones.

Operador	Descripción	Ejemplo	Resultado
&&	y	(x==10 && y==5) (x==10 && y>x)	true false
||	o	(x>=10 || y>x) (x<10 && y>x)	true false
!	no combinados	! (x==y) ! (x>y) (x>=10 && y<x || x==y) ((x<y || x>=10) && y>=5) (! (x==y) && y>=10)	true false true true false
&&	y	(x==10 && y==5) (x==10 && y>x)	true false

Operadores de comparación de JavaScript: resultados basados en valores iniciales de x=10 e y=5.

Fuente

- **Brad Dayley,** *Sams Teach Yourself NoSQL with MongoDB in 24 Hours.* Pearson Education, 2015.



educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!