MongoDB Fundamentos

Módulo 3

JavaScript en MongoDB

Uso de sentencias if

La sentencia **`if`** permite **separar la ejecución del código en función de la evaluación de una comparación.** Las siguientes líneas de código muestran la sintaxis. Los operadores condicionales se escriben entre paréntesis, y el código a ejecutar, si el condicional se evalúa como `true`, se indica entre corchetes (`{}`):

```
if(x==5){
  hacer_algo();
}
```

Además de ejecutar el código que se encuentra dentro del bloque **`if`** de instrucciones, se puede especificar un bloque **`else`** que se ejecute solo si la condición es `false`.

Por ejemplo:

```
if(x==5){
  hacer_algo();
} else {
  hacer_algo_else();
}
```

También es posible encadenar ó anidar sentencias **`if`**. Para hacer esto, se agrega una declaración condicional junto con una declaración **`else`**.

Por ejemplo:

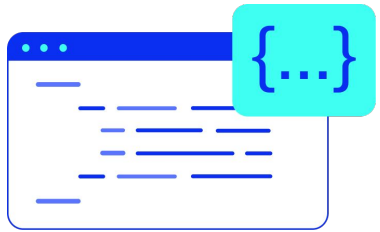
```
if(x<5){
  hacer_algo();
} else if(x<10) {
  hacer_algo_else();
} else {
  hacer_algo();
}
```

Implementación de sentencias switch

Otro tipo de lógica condicional es la sentencia **`switch`**. Permite evaluar una expresión una vez y luego, según el valor, ejecutar una de las muchas secciones diferentes de código. La sintaxis es la siguiente:

```
switch(expresión){
  case value1:
    <código a ejecutar>
    break;
  case value2:
    <código a ejecutar>
    break;
  default:
    <código a ejecutar si no se cumplen value1 o value2>
}
```

La instrucción **`switch`** evalúa la expresión por completo y obtiene un valor. Ese valor puede ser una cadena, un número, un valor booleano o incluso un objeto. Luego, la expresión **`switch`** se compara con cada valor especificado por la instrucción **`case`**. Si el valor coincide, se ejecuta el código de la sentencia. Si ningún valor coincide, se ejecuta el código predeterminado.



Implementación de bucles

El bucle es un medio que permite ejecutar el mismo segmento de código varias veces. Esto es extremadamente útil cuando se necesitan realizar las mismas tareas en una matriz o conjunto de objetos. JavaScript proporciona funcionalidad para realizar bucles **`for`** y **`while`**.

Bucles while

Es el bucle más básico en JavaScript. Un ciclo **`while`** prueba una expresión y continúa ejecutando el código contenido entre **`{}`** corchetes hasta que la expresión se evalúe como **`false`**.

Por ejemplo, el siguiente bucle **`while`** se ejecuta hasta que el valor de **`i`** es igual a 5:

```
var i = 1;
while (i<5){
  print("Iteración" + i + "\n");
  i++;
}
```

La salida resultante que aparecerá en la consola:

```
Iteración 1
Iteración 2
Iteración 3
Iteración 4
```

Bucles do / while

Otro tipo de bucle **`while`** es el bucle **`do/while`**. Se usa cuando se desea ejecutar el código contenido dentro del ciclo, al menos, una vez. Es decir, la expresión no se puede probar hasta que el código se haya ejecutado al menos una vez.

Por ejemplo, el siguiente bucle **`do/while`** se ejecuta hasta que el valor de `day` es igual a `miércoles`.

```
var dias = ["lunes", "martes", "miércoles",
            "jueves", "viernes"];
var i=0;
do{
  var dia=dias[i++];
  print("Es" + dia + "\n");
} while (dia != "miércoles");
```

La salida resultante por consola es:

```
Es lunes
Es martes
Es miércoles
```

Bucle for

Permite ejecutar código una cantidad específica de veces mediante una declaración **for** que combina tres declaraciones en un solo bloque de ejecución utilizando la siguiente sintaxis:

```
for (asignación; condición; actualización;){
  código a ejecutar;
}
```

La sentencia **for** utiliza esas tres declaraciones, al ejecutar el ciclo, como se muestra a la derecha:

- **Asignación**: se ejecuta una única vez, antes de que comience el bucle. De esta manera se inicializan las variables utilizadas en el ciclo como condicionales.
- **Condición**: se evalúa antes de cada iteración del ciclo. Si la expresión se evalúa como `true`, se ejecuta el ciclo; de lo contrario, **for** finaliza la ejecución del bucle.
- **Actualización**: ejecutó cada iteración después de que se haya ejecutado el código en el ciclo. Su uso típico es incrementar un contador usado en la instrucción 2.

Este ejemplo no solo ilustra un bucle básico **for**, sino que también muestra la capacidad de anidar un bucle dentro de otro:

```
for (var x=1; x<=3; x++){
  for (var y=1; y<=3; y++){
    print(x + " X " + y + " = " + (x*y) + "\n");
  }
}
```

La salida resultante por consola es:

```
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
```

Bucle for in

Otro tipo de bucle **for** es el bucle **for/ in**, que se ejecuta en cualquier tipo de datos que se pueda iterar. En su mayor parte, se usa el bucle **for/ in** en matrices y objetos. El siguiente ejemplo ilustra la sintaxis y el comportamiento del ciclo **for/ in** en una matriz simple:

```
var dias = ["lunes", "martes", "miércoles",
            "jueves", "viernes"];
for (var idx in dias){
  print("Es " + dias[idx] + "\n");
}
```

La variable `idx` se ajusta en cada iteración a través del bucle desde el índice de matriz inicial hasta el último. La salida resultante es:

```
Es lunes
Es martes
Es miércoles
Es jueves
Es viernes
```

Bucle con método foreach

Este método entra dentro del grupo de Iterables sin devolver una nueva matriz, lo que hace el **forEach** es ejecutar una función por cada elemento del arreglo. En cada iteración se tendrá acceso a 3 variables: valor (del elemento), índice (del elemento) y arreglo (que se está recorriendo).

Este método es muy útil cuando solo se necesita ejecutar una función a través de cada elemento del arreglo, sin necesidad de obtener un retorno.

```
const cars = ['Ferrari 250 GT
Berlinetta.', 'Tesla S', 'Génesis G90',
'Porsche Boxster'];
 //Con una función de devolución ES5
cars.forEach(function (element) {
  console.log(element);
});
```

```
//output Ferrari 250 GT Berlinetta
//output Tesla S
//output Génesis G90
//output Porsche Boxster
```

Fuente

Sams Teach Yourself NoSQL with MongoDB in 24 Hours - Brad Dayley - Pearsen Education 2015

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!