**GROUP BY**

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns. Example

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;

**setTimeout & setInterval**

The **setTimeout** & **setInterval** are the 2 timers functions of JavaScript. They can be used to create **functions** that executes **1 time** or **unlimited times**. That is, if you want a function to execute 1 time then use .setTimeout(), if you want to execute a function unlimited times then use .setInterval().

You can also add time in mill-seconds to these 2 function. That means - that milli-second is the time for them to execute after.

You can use the **“[setTimeout() and setInterval()](http://www.yogihosting.com/settimeout-setinterval-functions/" \t "_blank)”** functions on every JavaScript libraries and functions - jQuery, Angular, React, Vue, Knockoutjs, AngularJS, ….

*1. I want to show an alert just one time after 3 seconds.*

Here I will use **.setTimeout()** and the code will be:

setTimeout(function(){ alert("Hi"); }, 3000);

*2. I want to show an alert unlimited times after every 3 seconds.*

Here I will use **.setInterval()** and the code will be:

setInterval(function(){ alert("Hi"); }, 3000);

For example:   
setTimeout(function() { alert('foo');}, 1000);   
- waits for 1000 milliseconds, then runs the callback function (alerts 'foo') in this example   
  
setInterval(function() { alert('foo');}, 1000);  
- triggers callback every 1000 milliseconds. Meaning you'd get 'foo'-alerts every 1000 milliseconds, until you call the clearInterval-function to stop it.  
  
More info about JS timers:

**ORDER BY**

The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.

## Syntax

The basic syntax of the ORDER BY clause is as follows −

SELECT column-list

FROM table\_name

[WHERE condition]

[ORDER BY column1, column2, .. columnN] [ASC | DESC];

**return statement**

When a return statement is used in a function body, the execution of the function is stopped. If specified, a given value is returned to the function caller. For example, the following function returns the square of its argument, x, where x is a number.

function square(x) {

return x \* x;

}

var demo = square(3);

// demo will equal 9

If the value is omitted, undefined is returned instead.

The following return statements all break the function execution:

return;

return true;

return false;

return x;

return x + y / 3;

### Automatic Semicolon Insertion

### [Section](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/return#Automatic_Semicolon_Insertion)

The return statement is affected by [automatic semicolon insertion (ASI)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar#Automatic_semicolon_insertion). No line terminator is allowed between the return keyword and the expression.

To return a value other than the default, a function must have a [return](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/return) statement that specifies the value to return. A function without a return statement will return a default value. In the case of a [constructor](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/constructor) called with the [new](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new) keyword, the default value is the value of its this parameter. For all other functions, the default return value is [undefined](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/undefined).

### ****JavaScript Return Statement****

JavaScript provides for passing one value back to the code that called it after everything in the function that needs to run has finished running.

JavaScript passes a value from a function back to the code that called it by using the return statement. The value to be returned is specified in the return. That value can be a [constant value](https://www.thoughtco.com/using-constants-2034317), a variable, or a calculation where the result of the calculation is returned. For example:

return 3;  
return xyz;  
return true;  
return x / y + 27;​You can include multiple return statements into your function each of which returns a different value. In addition to returning the specified value the return statement also acts as an instruction to exit from the function at that point. Any code that follows the return statement will not be run.  
function num(x, y) {  
if (x !== y) {return false;}  
if (x < 5) {return 5;}  
return x;  
}

## **What are Cookies?**

Cookies are data, stored in small text files, on your computer.

When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.

Cookies were invented to solve the problem "how to remember information about the user":

* When a user visits a web page, his name can be stored in a cookie.
* Next time the user visits the page, the cookie "remembers" his name.

Cookies are saved in name-value pairs like:

username = John Doe