Each position in the table and the two-dimensional array is referred to by the column and row. In this table, Sammy is in position (0,0). Yellow is in position (0,3). Max is in position (3,0). Brown in in position (3,3). The first position is the column. The second position is the row. In PHP, [ ] are used to define the position (subscript) for an array.

```
echo $dogs[0][0] // displays Sammy
echo $dogs[0][3] // displays Yellow
echo $dogs[3][0] // displays Max
echo $dogs[3][3] // displays Brown
```

You can now adjust the loop to place the log contents in a two-dimensional array. However, you will not know the size of the array. So you can't use the format previously shown. This might cause developers to get a major migraine if they were not using PHP. PHP, however, allows you to dynamically create the array, just like it allows you to create variables (properties) whenever you need them.

```
$logFile = fopen("error_log.log", "r");
$row_Count = 0;
while(!feof($logFile))
{
        print_r ($error_Array[$row_Count] = explode(' | ', fgets($logFile)));
        $row_Count++;
}
fclose($logFile);
```

In the loop in this example, the `explode` method breaks the incoming line from the text file via the | character (actually a space, |, and a space). It places each separated string into the `$error_Array` at the row indicated by the value in `$row_Count`. The first time through the loop, the first line of the log file is placed in `$error_Array[0]` (the first row of the array). Because the `explode` command separated the string, this causes columns to be created for each piece.

If the first line of the file contained:

```
A general error | stuff | more stuff
```

then the first row of the array would contain:

```
$error_Array[0][0] = "A general error"
$error_Array[0][1] = "stuff";
$error_Array[0][2] = "more stuff";
```

You can verify this by using the `print_r` command shown in the example. `print_r` displays the contents of an array in the following format.

```
Array ( [0] => A general error [1] => stuff [2] => more stuff )
```

This format verifies that each piece of the string has been placed into the proper position in the array.

`$row_count` is incremented by 1 before the loop continues. This positions the next line of the file to be placed into the next position in the array (`$error_Array[1]`, if it is the second line of the file). You, of course, don't want to use `print_r` to display the results to the users (it's not very pretty).

However, it is a great tool to help you make sure the program is placing everything properly in the array. You can add code to the loop to build a table.