Loading JavaScript Arrays with MySQL Data

By Alex Ressi

Table of Contents

Introduction &Explaination

We have all seen pages that use JavaScript for better or for worse. In many cases JavaScript can improve a site's functionality and ease of use. Unfortunately administrating some of the complicated arrays that JavaScript depends on for things like heirarchichal menus and dynamic forms can be a pain in the rear. That's why were going to turn the task over to PHP and MySQL. We can use this combination to load data into the JavaScript for us. This is particularly useful if information contained in the array is likely to change.

In this exercise we will build a selection component for a resource management system. The component will tie people and project together based on staffing needs and employee skill. It will also illustrate how PHP and MySQL can be used to dynamically build JavaScript. The static component code is below.

Use the drop down menu below to select the skills required for the project. The list of personnel will change according to skill. Use the arrows arrows to control the addition or subtraction or people to the project.

This component uses two popular JavaScripts which are readily avialable on the web. I grabbed the JavaScript for the 'menu swapper' from www.javascriptsource.com, and I picked up a script to handle the drop down menu change from www.webreference.com.. With a little time, I managed to get the two scripts to work together as planned. View the source to see the resulting code. One of the first things you will notice is the following JavaScript array.

```
var ar = new Array();

ar[0] = new Array();
ar[0][0] = new makeOption("Crown, Tom", "151");
ar[0][1] = new makeOption("Christiansen, Steve", "221");
ar[0][2] = new makeOption("Berman, Randal", "321");
ar[0][3] = new makeOption("Turok, Steve", "341");
ar[0][4] = new makeOption("Cider, Eric", "361");
ar[0][5] = new makeOption("Bolton, Liz", "421");
ar[1] = new Array();
ar[1][0] = new makeOption("Crown, Tom", "152");
ar[1][1] = new makeOption("Christiansen, Steve", "222");
ar[1][2] = new makeOption("Berman, Randal", "322");
ar[1][3] = new makeOption("Turok, Steve", "342");
ar[1][4] = new makeOption("Cider, Eric", "362");
ar[1][5] = new makeOption("Bolton, Liz", "422");
ar[1][6] = new makeOption("Tuti, Berna", "432");
ar[1][7] = new makeOption("Dong, Enormai ", "442");
ar[2] = new Array();
```

```
ar[2][0] = new makeOption("Lindberg, John", "273");
ar[2][1] = new makeOption("Tuti, Berna ", "433");
ar[2][2] = new makeOption("Dong, Enormai", "443");
ar[3] = new Array();
ar[3][0] = new makeOption("Tuti, Berna ", "434");
ar[4] = new Array();
ar[4][0] = new makeOption("Narsysus, Thelma", "306");
ar[5] = new Array();
ar[5][0] = new makeOption("Turok, Steve ", "347");
ar[5][1] = new makeOption("Bolton, Liz ", "427");
```

The above code will serve as a model while we write our PHP code. Let's take a quick look at the anatomy of an array. The first set of brackets, ar[x], in this multi−dimentional array refers to the skill. The second set of brackets ar[x][x] is the array index of the item, which will always begin by default with 0. The item in this case is the employee. This array will be replaced by PHP code which will dynamcally build it. Now that we have played around with the component and had a look at the source code, it would be a good idea to build and populate that database.

Once the database has been built and populated, we need to do the following things to make our JavaScript dynamic. Note: The only portion of the source code that will be dynamic is the array, the rest of the JavaScript will remain static.

1. The database needs to be queried for employee names, and employee skills (two separate tables). The results need to be ordered by skill.
2. We will then need to loop through the skills printing the employee names associated with the skill
3. A mechanism then needs to be built to pass the employee id, skill id and project id to the form processing component.

Let's begin with the query. Have a look at the database schema to see how the information is stored. There are 3 tables involved in this component. Personnel, Skill, and person_skill.

```
$db = mysql_connect("localhost", "root", "");
mysql_select_db("extranet",$db);
```

A link to the database server is established, and the database is selected.

```
$sql = "SELECT
p.person_id,
s.person_id,
CONCAT(last_name,', ',first_name) AS name,
skill_id";
```

```
$sql .= "FROM
personnel p,
person_skill s
WHERE
p.person_id = s.person_id
ORDER BY
skill_id, name";

$result = mysql_query($sql);
```

The SQL statement is pretty straightforward. If you are unsure about what is going on here, you can always go to the MySQL site where there are numerous tutorials. The important thing to note in this query is the ORDER BY clause, which will properly setup the arrangement of the resulting data. After performing our SQL we then initialize two variables:

```
$type = "";
$number2 = "0";
```

We then will perform the while loop which will actually build the JavaScript array.

```
while ($myrow = mysql_fetch_row($result)) {
```

A series of "If then" statements will control the proper formation of the array.

```
if ($myrow[3] != $type) {
```

The first if statement checks to see if the variable $myrow[3] which is the skill_id from our SQL statement, is NOT equal to the variable $type. $type was set outside of the loop to nothing. The two values are not equal, so the next expression will be evaluated.

```
if ($number2 != NULL) {
```

$number2, which was initialized outside of the loop with a value of 0 is not equal to NULL, so the code within the curly braces gets run.

```
$newnumber2 = ($number2 + "1");
print ("ar[$number2] = new Array();\n");
$number2 = $newnumber2;
$type = $myrow[3];
$number = "0";
}
}
```

We have a new variable to start with, $newnumber2 which is given a value of 1. (0 + 1 = 1) The first line of the JavaScript array is then printed. ar[0] = new Array();

$number2 which was initially set to 0, now takes on the value of $newnumber2 which is 1. $type now is given a value. Initally set with no value and now $type has the value of $myrow[3] which is 0.

```
print "ar[" . ($number2 - "1") . "]";
if ($number != NULL) {
$newnumber = ($number + "1");
print ("[$number]");
$number = $newnumber;
}
```

From this code block we get the first part of the next line, namely ar[0][0]. The first '[0]' refers to the skill, so it will be repeated for each person that is associated with that particular skill. The next '[0]' refers to an individual possessing the skill. There is an "if statement." that increments the number in the second set of square brackets for each row in the database.

```
print (" = new makeOption(\"$myrow[2]\",
\"$myrow[1]$myrow[3]\");\n");
}
```

Before closing the while loop, we are going to append "= new makeOption("Crown, Tom", "151");" to the "ar[0][0]", thus completing one pass through the loop. The loop will be run for each row in the database query, which is in this case is 21. You can view the entire unbroken source code here. The next challenge will be passing multiple values to the form processing script. This will be done using a combination of JavaScript and PHP, and will be the focus of a seperate upcoming article.

In addition to building JavaScript arrays, this code can be hacked up for a number of other uses . What this

code essentially does is print the first row of a given field, where rows in the database have fields with common values. In this example, we initialize an array which represents a given skill, for instance ar[0] = new Array();. Under that we print all the people in this case that relate to that skill. This can easlily be applied to something like a classified system. Let's say you are selling automobiles and you would like to print as a header, the make of the car before listing all the models that fall under it. You could use this code to do the same. There are many uses for this code. Don't be afraid to try things out. The world is now your oyster, enjoy.

Source Reference

Plug this in place of the JavaScript array in the source code of the refering page and go! PHP can be inbeded in JavaScript tags.

```
<?php
$db = mysql_connect("localhost", "root", "");
// This establishes a link to MySQL
mysql_select_db("extranet",$db); // The database is specified

$sql = "SELECT
p.person_id,
s.person_id,
CONCAT(last_name,', ',first_name) AS name,
skill_id ";

$sql .= "FROM
personnel p,
person_skill s
WHERE
p.person_id = s.person_id
ORDER BY
skill_id, name";

$result = mysql_query($sql);

$type = "";
$number2 = "0";
while ($myrow = mysql_fetch_row($result)) {
if ($myrow[3] != $type) {
if ($number2 != NULL) {
$newnumber2 = ($number2 + "1");
print ("ar[$number2] = new Array();\n");
$number2 = $newnumber2;
$type = $myrow[3];
$number = "0";
}
}
print "ar[" . ($number2 - "1") . "]";
if ($number != NULL) {
$newnumber = ($number + "1");
print ("[$number]");
$number = $newnumber;
}
print (" = new makeOption(\"$myrow[2]\",
\"$myrow[1]$myrow[3]\");\n");
```

```
}
?>
```

The drop down menu with skills is also database driven so that new skills can easily be added to the database. Here is the code that was used to generate it.

```
<SELECT NAME="industry" onChange="relate(this.form)">
<?
$db = mysql_connect("localhost", "root", "");
mysql_select_db("extranet",$db);

$sql2 = "SELECT DISTINCT
s.skill_id,
p.skill_id,
skill_name ";

$sql2 .= "FROM
skill s,
person_skill p
WHERE
s.skill_id = p.skill_id
ORDER BY
s.skill_id";

$result2 = mysql_query($sql2);

while ($myrow2 = mysql_fetch_row($result2)) {
print ("
```

The following is the code to build and populate the the tables that are used in this module. It can be cut out of the web page and then pasted into a text file on your database server where it can then be imported by MySQL using the mysqlimport command.

```
#
# Table structure for table 'personnel'
#
CREATE TABLE personnel (
person_id int(11) DEFAULT '0' NOT NULL auto_increment,
first_name varchar(15),
last_name varchar(15),
company varchar(30),
PRIMARY KEY (person_id)
);

#
```

```
# Dumping data for table 'personnel'
#

INSERT INTO personnel (person_id, last_name, first_name,
company) VALUES (34,'Turok','Steve','1');
INSERT INTO personnel (person_id, last_name, first_name,
company) VALUES (32,'Berman','Randal','1');
INSERT INTO personnel (person_id, last_name, first_name,
company) VALUES (30,' Vi
jaya','Narayanas','1');
INSERT INTO personnel (person_id, last_name, first_name,
company) VALUES (27,' Jo
han','Lindgren','1');
INSERT INTO personnel (person_id, last_name, first_name,
company) VALUES (22,'Christiansen','Steve','1');
INSERT INTO personnel (person_id, last_name, first_name,
company) VALUES (15,'Crown','Tom','1');
INSERT INTO personnel (person_id, first_name, last_name,
company) VALUES (36,'Cider','Eric','1');
INSERT INTO personnel (person_id, first_name, last_name,
company) VALUES (42,'Bolton','Liz','1');
INSERT INTO personnel (person_id, first_name, last_name,
company) VALUES (43,'Tuti','Berna','1');
INSERT INTO personnel (person_id, first_name, last_name,
company) VALUES (44,'Dong','Enormai','1');



#
# Table structure for table 'person_skill'
#
CREATE TABLE person_skill (
person_id int(11) DEFAULT '0' NOT NULL,
skill_id tinyint(2),
level tinyint(1)
);

#
# Dumping data for table 'person_skill'
#

INSERT INTO person_skill (person_id, skill_id, level)
VALUES (15,1,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (15,2,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (22,1,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (22,2,NULL);
```

```
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (27,3,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (30,6,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (32,1,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (32,2,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (34,1,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (34,2,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (34,7,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (36,1,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (36,2,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (42,1,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (42,2,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (42,7,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (43,4,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (43,2,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (43,3,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (44,2,NULL);
INSERT INTO person_skill (person_id, skill_id, level)
VALUES (44,3,NULL);

#
# Table structure for table 'skill'
#
CREATE TABLE skill (
skill_id int(11) DEFAULT '0' NOT NULL auto_increment,
skill_name varchar(20),
skill_desc varchar(250),
PRIMARY KEY (skill_id)
);

#
# Dumping data for table 'skill'
#
```

```
INSERT INTO skill (skill_id, skill_name, skill_desc)
VALUES (6,'Oracle',NULL);
INSERT INTO skill (skill_id, skill_name, skill_desc)
VALUES (5,'ASP',NULL);
INSERT INTO skill (skill_id, skill_name, skill_desc)
VALUES (4,'Cold Fusion',NULL);
INSERT INTO skill (skill_id, skill_name, skill_desc)
VALUES (3,'Vignette',NULL);
INSERT INTO skill (skill_id, skill_name, skill_desc)
VALUES (2,'JavaScript',NULL);
INSERT INTO skill (skill_id, skill_name, skill_desc)
VALUES (1,'HTML',NULL);
INSERT INTO skill (skill_id, skill_name, skill_desc)
VALUES (7,'MySQL',NULL);
```