Jquery Tutorial: Passing php arrays to javascript arrays - Developer News

NEWS_PDF_AUTHOR: kaotik

NEWS_PDF_DATE: 2009/11/12 16:30:00

Passing php arrays to javascript arrays. This is an advanced tutorial designed for people who are comfortable using jquery, ajax and php. It will teach you how to retrieve a php array using jquery, convert it to a javascript array then manipulate it.

I will show you two distinct methods of accomplishing the same result; 1 using JSON through jquery and the other a personal hack that also gets the job done.

Step 1 Understanding the dilemma

Arrays are great when we want to pass several pieces of information using just 1 variable. When creating a php array you can use this format:

```
 $list['id'] = '3';
$list['name'] = 'jack in a box';
$list['desc'] = 'bla bla bla and more bla';
```

Here we have an associative array with 3 elements. An associative array uses names for it's items; $list['id']. Javascript does NOT support associative arrays so we will have to use this format:

```
 $list[0] = '3';
$list[1] = 'jack in a box';
$list[2] = 'bla bla bla and more bla';
```

Note: javascript arrays always start with 0.

METHOD 1 - My own method

This method really isn't elegant as JSON but it does get the job done.

Step 2- Objective
Most of you can dynamicly generate a list from the database. So my objective is to use a list that, when an element is clicked on, jquery will load all info regarding that element and populate a form. This is a great method for your users to edit info without having to reload the page.

Step 3- Creating the HTML
Create an html page called test.html and place this code inside:

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

});
```

cool site
new name
great space

Add New

Name
Desc

As you can see, this is a simple html page with a list and a form. Based loosly on my previous tutorials. The 'div' ajaxBox will hold the server response to the clicks. You will also notice on the list that I have an attribute called 'myval'. This holds the id for each element and is used with jquery.
Now create an empty php file called myserv.php.

Step 3- Jquery clicks

The next step is setup an ajax response to each item list click.


```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

$("#mylist a").bind("click", function() {
var hol=$(this).attr('myval');
var formContent ="action=getlink&link="+hol;
$("#ajaxBox").load('myserv.php',formContent);
});

});
```

cool site
new name
great space

Add New

Name
Desc

Now open myserv.php and replace all code with this:

I've covered this code in previous tutorials so I won't go indepth on what this does.

So now we have info being passed to the server:

var formContent ="action=getlink&link="+hol;

and the server response gets placed inside "#ajaxBox". Great! Now letsuse the id to grab our

data. Replace all code inside myserv.php with this:

We now have a function that loads all info. To keep this tutorial simple I used a switch to alternate between 3 examples. Look at this line:
$str=$list[0].''.$list[1];

This is the format I'm using to later pass this into a javascript array. The php variable should be sent back as a string. It also needs a concate symbol that absolutly WON'T be used anywhere else, that's why I came up with .

Reload the html page any try clicking on the list items. You can see our new string now inside ajaxBox.

Step 4 - Setting javascript to accept our php string.
Javascript has a nice function called 'split'. It takes a string, splits it acording to a defined delimiter, then places each piece in a javascript array. Open test.html and replace all code with this:

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

$("#mylist a").bind("click", function() {
var hol=$(this).attr('myval');
var formContent ="action=getlink&link="+hol;
$("#ajaxBox").load('myserv.php',formContent);

var txt = $("#ajaxBox").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
});

});
```

cool site
new name
great space

Add New

Name
Desc

There's only 4 new lines that need explaining:

```
var txt = $("#ajaxBox").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
```

The first line grabs all text inside our div "#ajaxBox". This is an intermediate step that we simply can't avoid using this method. I will show you a clever way of making this nicer further on.
The second line splits our string acording to the delimiter "" and places it in a javascript array called 'php'.
The third and forth line each grab a piece of the javascript array and assign it to each form element.

So now, when a user clicks on a list item, jquery does an ajax call, grabs info from the server and places it inside the form.It's a great way to allow users to edit info!

Now try clicking further. Notice theirs a discrepenece between what appears in "ajaxBox" and the form. This is because javascript is being processed faster then the elements in the page get refreshed. To fix this we will place a small delay in our code. Download this jquery plugin. Now open test.html and replace this:

with this:

Replace this:

```
 var txt = $("#ajaxBox").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
```

with this:

```
 $(this).delay(500,function(){
var txt = $("#hidden").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
});
```

So now we have a jquery function that delays execution for 1/2 a second, it then executes the lines inside. Try testing it now. You can adjust the delay timer to your liking by changing '500'.

Step 5- Making it all look nice.
Although this does work OK, The only text I want apearing in "ajaxBox" should be server alerts or messages. So I'm going to create a hidden div that will serve as an intermediate step.

in test.html replace all code with this:

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

$("#hidden").hide();
$("#textfield").val("");
$("#textarea").val("");

$("#mylist a").bind("click", function() {
var hol=$(this).attr('myval');
```

```
var formContent ="action=getlink&link="+hol;
$("#hidden").load('myserv.php',formContent);

$(this).delay(500,function(){
var txt = $("#hidden").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
$("#formHeader").text("Edit");
$("#ajaxBox").text("All info loaded OK");
});
});

});
```

cool site
new name
great space

Add New

Name
Desc

I've added a couple of new things. At the beginning of javascript code, I now clear any info inside form elements. When you click on a list element the title of the form is changed from 'Add New' to 'Edit'. A message also apeears in the ajaxBox with a 'all OK'.
Like I've said before, this method is neither good nor bad. It works. The JSON method I'll show you next week is probably the preferred method since it's more straightforward to apply and is directly supported in jquery.
I hope you've enjoyed this tutorial and stay tuned for the **JSON method**!

Passing php arrays to javascript arrays. This is an advanced tutorial designed for people who are comfortable using jquery, ajax and php. It will teach you how to retrieve a php array using jquery, convert it to a javascript array then manipulate it.

I will show you two distinct methods of accomplishing the same result; 1 using JSON through jquery and the other a personal hack that also gets the job done.

Step 1 Understanding the dilemma

Arrays are great when we want to pass several pieces of information using just 1 variable. When creating a php array you can use this format:

```
 $list['id'] = '3';
$list['name'] = 'jack in a box';
$list['desc'] = 'bla bla bla and more bla';
```

Here we have an associative array with 3 elements. An associative array uses names for it's items; $list['id']. Javascript does NOT support associative arrays so we will have to use this format:

```
 $list[0] = '3';
$list[1] = 'jack in a box';
$list[2] = 'bla bla bla and more bla';
```

Note: javascript arrays always start with 0.

METHOD 1 - My own method

This method really isn't elegant as JSON but it does get the job done.

Step 2- Objective
Most of you can dynamicly generate a list from the database. So my objective is to use a list that, when an element is clicked on, jquery will load all info regarding that element and populate a form. This is a great method for your users to edit info without having to reload the page.

Step 3- Creating the HTML
Create an html page called test.html and place this code inside:

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

});
```

```
cool site
new name
great space
```

```
Add New

Name
Desc
```

As you can see, this is a simple html page with a list and a form. Based loosly on my previous tutorials. The 'div' ajaxBox will hold the server response to the clicks. You will also notice on the list that I have an attribute called 'myval'. This holds the id for each element and is used with jquery.
Now create an empty php file called myserv.php.

Step 3- Jquery clicks

The next step is setup an ajax response to each item list click.

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript
```

```
$("#mylist a").bind("click", function() {
var hol=$(this).attr('myval');
var formContent ="action=getlink&link="+hol;
$("#ajaxBox").load('myserv.php',formContent);
});

});
```

cool site
new name
great space

Add New

Name
Desc

Now open myserv.php and replace all code with this:

I've covered this code in previous tutorials so I won't go indepth on what this does.

So now we have info being passed to the server:

var formContent ="action=getlink&link="+hol;

and the server response gets placed inside "#ajaxBox". Great! Now letsuse the id to grab our data. Replace all code inside myserv.php with this:

We now have a function that loads all info. To keep this tutorial simple I used a switch to alternate between 3 examples. Look at this line:
$str=$list[0].".$list[1];

This is the format I'm using to later pass this into a javascript array. The php variable should be sent back as a string. It also needs a concate symbol that absolutly WON'T be used anywhere else, that's why I came up with .

Reload the html page any try clicking on the list items. You can see our new string now inside ajaxBox.

Step 4 - Setting javascript to accept our php string.
Javascript has a nice function called 'split'. It takes a string, splits it acording to a defined delimiter, then places each piece in a javascript array. Open test.html and replace all code with this:

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

$("#mylist a").bind("click", function() {
var hol=$(this).attr('myval');
var formContent ="action=getlink&link="+hol;
$("#ajaxBox").load('myserv.php',formContent);

var txt = $("#ajaxBox").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
});

});
```

```
cool site
new name
great space
```

Add New

Name
Desc

There's only 4 new lines that need explaining:

var txt = $("#ajaxBox").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
The first line grabs all text inside our div "#ajaxBox". This is an intermediate step that we simply can't avoid using this method. I will show you a clever way of making this nicer further on.
The second line splits our string acording to the delimiter "" and places it in a javascript array called 'php'.
The third and forth line each grab a piece of the javascript array and assign it to each form element.

So now, when a user clicks on a list item, jquery does an ajax call, grabs info from the server and places it inside the form.It's a great way to allow users to edit info!

Now try clicking further. Notice theirs a discrepenece between what appears in "ajaxBox" and the form. This is because javascript is being processed faster then the elements in the page get refreshed. To fix this we will place a small delay in our code. Download this jquery plugin. Now open test.html and replace this:

with this:

Replace this:

```
 var txt = $("#ajaxBox").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
```

with this:

```
 $(this).delay(500,function(){
var txt = $("#hidden").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
});
```

So now we have a jquery function that delays execution for 1/2 a second, it then executes the lines inside. Try testing it now. You can adjust the delay timer to your liking by changing '500'.

Step 5- Making it all look nice.
Although this does work OK, The only text I want apearing in "ajaxBox" should be server alerts or messages. So I'm going to create a hidden div that will serve as an intermediate step.

in test.html replace all code with this:

```
#ajaxBox { background-color:#FFFF99; border:thin solid #FF0000; width:70%; height:50px;}
#formHeader{text-align:center; font-size:18px; color:#0000FF;}
#myform{text-align:center;}
```

```
$(document).ready(function() { //Finish loading the entire page before processing any javascript

$("#hidden").hide();
$("#textfield").val("");
$("#textarea").val("");

$("#mylist a").bind("click", function() {
var hol=$(this).attr('myval');
var formContent ="action=getlink&link="+hol;
$("#hidden").load('myserv.php',formContent);

$(this).delay(500,function(){
var txt = $("#hidden").text();
var php=txt.split("");
$("#textfield").val(php[0]);
$("#textarea").val(php[1]);
```

```
$("#formHeader").text("Edit");
$("#ajaxBox").text("All info loaded OK");
});
});

});
```

cool site
new name
great space

Add New

Name
Desc

I've added a couple of new things. At the beginning of javascript code, I now clear any info inside form elements. When you click on a list element the title of the form is changed from 'Add New' to 'Edit'. A message also apeears in the ajaxBox with a 'all OK'.
Like I've said before, this method is neither good nor bad. It works. The JSON method I'll show you next week is probably the preferred method since it's more straightforward to apply and is directly supported in jquery.
I hope you've enjoyed this tutorial and stay tuned for the **JSON method**!