How to pass Php variables and Arrays to JavaScript

by George Lunt

To pass a Php variable to a JavaScript variable is relatively simple. All you do is equate the print(or echo) of the Php variable to a JavaScript variable. The simple script below illustrates this technique:

Script one.php

```
<?php
   $testvar = "Hello World";
?>
<html>
<head>
<script type="text/javascript">
function hello()
 {
   // create JavaScript variable, fill it with Php variable
   var testvar = "<? print $testvar; ?>";
  // output to screen

   document.write( testvar );
 }
</script>
</head>

<!-- Call JavaScript function to display variable -->
<body onload="hello()" >
</body>
</html>
```

Notice that the above script printed "Hello World" using a JavaScript function, with a variable that was passed from the php server script. Now let's try the same thing with arrays.

Script two.php

```
<?php
   $testarray = array('zero', 'one', 'two', 'three', 'four');
?>
<html>
<head>
<script type="text/javascript">
function showarray()
 {
   var testarray = new Array();
   testarray = "<? print $testarray; ?>";
   //Display output of all the array elements;
   for( x=0; x<5; x++){
        document.write( testarray(x) + "<br>" );
   }
 }
</script>
</head>

<body onload="showarray()" >
</body>
</html>
```

If you attempt to run the script above, instead of outputting a display of the five array elements, we getting the word "undefined". We even created a JavaScript array expecting it to be filled with the Php array. Unfortunately, it looks like the array handling systems are not similar enough to accept this type of array passing.

But because we were able to pass a string variable from Php to JavaScript, why not convert the Php array to a string, pass the string to JavaScript, and finally convert the string to a JavaScript array. This is not as hard as it seems, because Php has the implode function and JavaScript has the split method. This code makes an array to string conversion a snap. In the code below, an array created by Php is downloaded from the server and converted to a JavaScript array.

Script three.php

```
<?php
   //create php test array
   $testarray = array('zero', 'one', 'two', 'three', 'four');
   //convert array to string using ":#:" as separator
   //using complex separator insures that it is not part of array element
   $testvar = implode(":#:",$testarray);
?>
<html>
<head>
<script type="text/javascript">
function showArray()
 {
   //Convert Php string to JavaScript string
   var testvar = "<? print $testvar; ?>";
  //Create JavaScript array
   var testarray = new Array();
   var x;
   //Fill JavaScript array from the converted string
   testarray = testvar.split(":#:");
   //Display output of all the array elements;
   for( x=0; x<5; x++){
        document.write( testarray[x] + "<br>" );
   }
 }
</script>
</head>

<!-- Call JavaScript function to display array element -->
<body onload="showArray()" >
</body>
</html>
```

If you run the above script, you see the output of all the array elements, as expected, indicating that we have successfully created a JavaScript array from a Php array.

This has some interesting implications. The Php section of the code is actually performed on the server, but the JavaScript part is run on the client computer. After the Php code runs sending all its variables to the client, the JavaScript can takeover and you can literally be disconnected from the server. For some applications, this can mean much less strain on the server than maintaining a server session. After you're done manipulating your variables with JavaScript, you can send your results back to the server.

To move JavaScript variables back to the sever, convert all arrays to strings using the JavaScript join method, then use the form method post(or get) to pass them back to the server within hidden variables. On the sever use the Php function explode() to convert your strings back to an array. Here is an example.

Script four.php

```
<?php
 $testvar = $_POST['testvar'];
 // Run this section only if $testvar was posted
 if ( $testvar ){
    // Convert passed variable to array
    $testarray = explode(":#:", $testvar );
    // Display values of array elements
    for ( $x=0; $x<5; $x++ ){
        print $testarray<$x>."<br>";
    }
 }else{
// Run the html script only if $testvar doesn't exist
?>
<html>
<head>
<script type="text/javascript">
function submitString()
 {
   //Create JavaScript array and fill it
   var testarray = new Array('zero', 'one', 'two', 'three', 'four');
   //Create a string from array
   var testvar = testarray.join(":#:");
   //Set string as the value of the hidden form element
   document.getElementById('testvar').value = testvar;
   return true;
 }
</script>
</head>

<body>
<form  method = "post" action = "<?  print $_SERVER['PHP_SELF']; ?>"
onsubmit = "return submitString();">
<input type = "hidden" id = "testvar"  name = "testvar" >

<input type = "submit" value = "Create JS Var and Submit" >
</form>
</body>
</html>
<?php
// Don't forget the final curly bracket to end the else construct
}
?>
```

On-line testing is an example of passing arrays from Php to JavaScript and visa versa. Other examples can be surveys, slideshows, e-books, or even shopping carts. If run by Php alone, such applications require server sessions, and can load the server if multiple users start sessions simultaneously.

The sample scripts all work, although they may not be very useful. Just cut the code and paste it into a Php file. Remember you need a Web Server with Php installed to test these samples.

George Lunt is someone who feels the world is getting too corporate. His writings relate to the individual's struggle with big government and big corporations. His website is http://www.corporate-aliens.com.

No add version of article for printing/downloading