Security and performance—In this application, the dogs information is not highly sensitive. Also, the information is displayed to the users. This allows the users to see and fix any data that might have been corrupted. For more sensitive information, the array should be declared using `var` *to keep the data accessable only to the current method (function).*

The subscripts (0, 1, and 2) can now be used to pull each part of the array and release it to the proper place. `responsevalues[0]` is placed in `AjaxResponse` to display the `getBreeds` list. `responsevalues[1]` is placed in `AjaxReturnValue` to display the dogs list box. The dogs array, since you won't know yet which dog the user has selected, will be placed in a JSON object (`obj`). As you have seen, JSON data includes named indexes and values, which are very similar to associate arrays in PHP. The `var` statement is not used because this object must be public and available to the complete application.

If you remember from previous chapters, arrays cannot be directly formatted into strings. Arrays must be serialized. However, JSON data can also be passed in a string. You will soon see that the "array" in `responsevalues[3]` has been formatted as JSON data by the `dog_interface` program. JavaScript's `JSON.parse` method has the ability to look at data and, if it is valid, transform it into a JSON object. This is very similar to the PHP method `json_encode`.

Now let's look at how you can populate the form when the user picks a dog from the dogs list box.

The JavaScript method `process_select` (called by the HTML button after the user picks from the dogs list) has been placed at the top of the code in the `lab.php` file. It could have also been placed in its own JS file and imported in the same way as the `getlists.js` file. This new method use the information contained in OBJ (the JSON dogs object containing all the dogs) to populate the text boxes (`dog_name` and `dog_weight`), radio buttons (`dog_color`), and list box (`dog_breed`) with the information for the dog the user selects in the list box.

```
function process_select() {
         var colorbuttons = document.getElementsByName('dog_color');
```

First, all the color values from the radio buttons will be pulled from the HTML form and placed into an array called `colorbuttons` using the JavaScript method `getElementsByName.` `dog_color` (the name of each radio button) is passed into the method. This process will create an array of the radio buttons with the same indexes as the radio button subscripts for the color. For instance, the `0` position of the array will now contain `brown`, which is the first radio button displayed in the HTML form. This will allow you to set the proper color radio button by referencing its position (such as `colorbuttons[0]` to set `brown`).

```
if(!(document.getElementById('dogs').value == -1))
{
         index = document.getElementById('dogs').selectedIndex -1;
         document.getElementById('index').value = index;
         document.getElementById('dog_name').value = obj.dogs[index].dog_name;
         document.getElementById('dog_weight').value = obj.dogs[index].dog_weight;
```

HTML list boxes include both text and values. The text is what the user sees; the value is what it represents. This is very similar to PHP associative arrays—keys (indexes) and values. The `if` statement checks the dogs array to determine its current value. If the value is `-1`, this indicates that the users did not select anything, or they selected NEW. The JavaScript `!` symbol works the same as the PHP `!` symbol. The symbol changes the `if` portion of the statement to execute when the value for `'dogs'` is not `-1`. Thus, the code will execute if the user has selected a dog from the dogs list box.

The `selectedIndex` property of a list box indicates the index selected by the user. However, the HTML list box is numbered starting at 1. JavaScript arrays and JSON objects indexes start at 0. This causes the `selectedIndex` to be one more than the position in a JavaScript array or JSON object. The code, in the example, subtracts 1 to balance out the relationship. This value is placed in index. It is also saved in the

www.it-ebooks.info