■ **Note** Different browsers react in different ways to JavaScript code. Internet Explorer tries to block JavaScript code by default. If you see no results and are sure your code is correct, yet are receiving error messages about missing methods (such as validate_input) it may be that your browser's ability to execute JavaScript is turned off. Either turn it on in the settings of the browser or try a different browser that allows JavaScript to run.

Security and performance—Although you have verified input on the HTML page, the data could still cause harm to the system. The user could try to send HTML, JavaScript, and even PHP code that could to cause harm. This could occur from bad information entered in the HTML page, a packet sniffing program intercepting the data transferring to the PHP page, or a program trying to bypass the HTML page altogether. No matter which of these activities produced the harmful information, the PHP program will need to detect it and handle the problem. Therefore, the receiving program (`lab.php`) must try to filter (clean and remove harmful code) the data.

Do It

1. Adjust Example 4-2 to validate the gender text box from the first Do It. Make sure to include an error message to indicate the problem(s). Also make sure the error message is added to the `$error_message` property to be displayed with all other error messages. Make sure to test your code for all possible problems with the user entering data. Don't forget to test if they do not enter any data.

PHP Filtering

It's time to look at the changes required to the php file (`lab.php`). You need to be able to accept in the properties and values (`dog_name`, `dog_breed`, `dog_color`, and `dog_weight`) from the HTML program. However, you need to be concerned that someone might try to send information that could affect the operation of the program or even crash the system in which it is operating.

This section takes two approaches to help reduce the possibility of harmful data. First, you will determine if you have received all required information. If not, you will ask the user to return to the HTML page (`lab.html`) to enter the information. This will, at least, make sure that you have all data and it meets the validation provided by the HTML5 page and/or JavaScript program (`validator.js`). Second, you will use some existing PHP methods to filter out HTML, JavaScript, and PHP syntax from the data received. This will reduce the chance that an executable statement will be passed into the program.

Example 4-3. Partial listing of the top of lab.php with the clean_input method

```
<?php
require_once("dog.php");
function clean_input($value)
{
$bad_chars = array("{", "}", "(", ")", ";", ":", "<", ">", "/", "$");
$value = str_ireplace($bad_chars,"",$value);
// This part below is really overkill because the string replace above removed special characters
$value = htmlentities($value); // Removes any html from the string and turns it into &lt; format
```