The Continue option restarts script execution. Execution will not end until either another breakpoint is encountered or the script completes. The Continue option is useful when you need to track down the location of a problem. You can set breakpoints at points along the execution chain and, each time a breakpoint is encountered, inspect variable values to see whether the problem has cropped up yet. Once the problem appears, you know the error occurred somewhere between the current breakpoint and the previous breakpoint, and you can narrow it down further from there. The Continue option is also useful when debugging an iteration. You can set a breakpoint at one point within the iteration and use the Continue option to speed through the iterated code, checking every time that execution suspends whether any problems have occurred.

The Step Over function is useful when you want to avoid stepping through a function that is called by the current function. The called function may be a function that has been extensively debugged and you just know the problem isn’t there, or you may just want to avoid stepping through its code because you’re concerned about the current function only. Keep in mind that stepping over a function does not prevent it from being executed; it merely means you’re not going to step through it line by line. Instead, the function will be executed as if it were one statement.

The Step Into option is the opposite of the Step Over function. Step Into will step into a called function so you can debug the called function. Step Over and Step Into work well together when you’re trying to track down the exact location of an error. Consider the situation where you’re debugging the function sendAjaxRequest. This function contains several lines of logic itself, and in addition sendAjaxRequest calls createXmlHttp. The first time you step through sendAjaxRequest, carefully step over each line and watch for when the error first manifests itself. When you encounter the breakpoint for createXmlHttp, use Step Over to quickly execute createXmlHttp without stepping through it line by line. After stepping over createXmlHttp, check to see whether the error has yet shown up. If not, then you know the error does not occur within createXmlHttp. If it does show up, then you know the error occurs within createXmlHttp, and the next time debugging through sendAjaxRequest, you should step into createXmlHttp and debug from there.

You can consider the Step Out option the “antidote” to Step Into. Step Out allows you to exit debugging of the current function and return to the previous function in the call stack. If you’re currently stepping through the function createXmlHttp, which was called from sendAjaxRequest, then using the Stop Out option will return to the line in sendAjaxRequest that follows the call to createXmlHttp. Step Out is particularly useful when you are in a function and don’t want to step completely through it and would rather return the previous function in the call stack.

By now you may have noticed the Stop button on Venkman’s toolbar and wondered what it does. Clicking the Stop button will activate Venkman’s ability to suspend JavaScript immediately the next time any JavaScript is executed. Once you have clicked the Stop button, you can be sure that the next time any JavaScript runs, Venkman will suspend its execution. Scripts that are run automatically using setTimeout or setInterval are often difficult to debug because the script’s entry point is difficult to determine. By activating the Stop function, Venkman will trap the script’s entry point, and you can debug from there, as needed, as shown in Figure 7-18.

Figure 7-18. Venkman’s Stop button
deactivated (left) and activated (right)