tdAutoSelectBox Documentation

Tetraktys Development

http://www.tetraktysdevelopment.com

Table of Contents

Introduction

Text boxes and select boxes have long been separate objects on web-based forms. The HTML language lacks any form object that is equivalent to Windows' more advanced combo boxes, which enable a user to type in the select box as well as choose from the drop-down menu.

tdAutoSelectBox (short for "Tetraktys Development Auto-Complete Select Box") is here to change that. Using advanced javascripting, tdAutoSelectBox allows a webmaster or Internet application designer to implement this highly customizable control. With tdAutoSelectBox, webmasters are no longer confined to the limitations of the traditional browser-rendered select box, enabling them to unleash the full potential of their web site, web portal, or Internet application. tdAutoSelectBox is an advanced javascript combo box, not only allowing the user to choose an item from the drop-down list, but also works as a textbox, enabling the user to quickly type the item name rather than scrolling through a list of items.

tdAutoSelectBox vs Select Box

What advantages does tdAutoSelectBox have over the traditional select box? First and foremost, tdAutoSelectBox is designed to fully replace the browser rendered select boxes and multi-select boxes. Apart from avoiding the various issues with the browser rendering of select boxes, tdAutoSelectBox adds the ability for a user to type inside the select box for faster entry. You may also further customize tdAutoSelectBox in ways that are impossible with browser-rendered select boxes and multi-select boxes.

Problems with Traditional Select Box

- **Not customizable**. Select boxes are rendered by the browser, and only a limited amount of customization is possible.
- **Not adjustable.** It is not possible to control the height of the drop-down box with the traditional select boxes, so you cannot limit the drop-down to show, for example, 5 items at a time.
- **Serious flaws.** Select boxes tend to "tear-through" floating page elements, such as DIV windows, javascript menus and floating calendars, and similar elements.
- **Hard to use.** The multi-select option on traditional select boxes is a "hack", and is not intuitive to use. How many users are going to control-click to select 6 different items in a long list?
- **Inadequate features.** Typing a letter only jumps to the first occurrence of that letter in a traditional select box. The user cannot type the full name of the item to instantly select it.
- **Cumbersome interface.** Large select box lists can take a minute or more to search. If it had the ability to auto-complete as the user types, the user could limit his search to a much smaller list.

Advantages of tdAutoSelectBox

- **Easy to use.** With just a few lines of code, you can replace your existing select boxes with AutoSelectBoxes.
- **Intuitive.** AutoSelectBox is an intuitive select box, allowing users to type the value they wish to select, just as with Windows-like combo boxes and auto-complete text boxes. Matching values are pre-selected as they are typed, and the user is free to select from the scrollable drop down list.
- **Multi-select.** The Multi-Select Box feature improves upon the default browser's

multi-select box.

- **Fixes browser issues.** Avoids "select box tear-through" problems inherent with the browser-rendered select boxes which can interfere with advanced javascript menus, calendars, and floating windows.
- **Adjustable.** Adjust the height of the drop-down list to any size you want -- no longer are you reliant on the browser's rendering engine.
- **Execute other Javascript.** *OnSelect* attribute allows external javascript functions to be executed when an item is selected in the AutoSelectBox, such as a pop-up alert, DOM changes, or any other javascript action.
- **AJAX capable.** New AJAX capability has been added for advanced web site developers. Now, tdAutoSelectBox can call-back to server-side scripts and process data dynamically.
- **Fully customizable.** tdAutoSelectBox is configured using CSS, so you can customize how it looks to fit with your site's design, including font size, colors, button styles, border styles, and more.

Getting Started with tdAutoSelectBox

Developing a web site or web application that makes use of tdAutoSelectBox is only marginally more difficult than using the traditional select boxes. Once you get the hang of it, creating or converting all of your select boxes to AutoSelectBoxes will be a snap.

First, if you have not already done so, you will need to purchase and download a licensed copy of tdAutoSelectBox from Tetraktys Development. Please visit http://www.tetraktysdevelopment.com

Supported Browsers

tdAutoSelectBox is designed to work on all major modern web browsers. The following browsers are officially supported.

- Internet Explorer 5.5, 6.0+
- Mozilla Firefox 1.0+, 1.5+
- Mozilla Suite 1.5+
- Netscape 7.2+, 8.0+
- Opera 8.5+

Although tdAutoSelectBox may work on other unsupported browsers (such as Safari) we make no guarantee of this. AJAX functions and *OnSelect* attribute will not work IE 5.0 or Opera 7, or earlier versions of Netscape and Mozilla Suite, due to technical limitations.

Installing tdAutoSelectBox on your web site

tdAutoSelectBox comes in a ZIP file archive. Unzip this file to create the tdAutoSelectBox folder, under which you will find various other files and folders. Copy or FTP the tdAutoSelectBox folder to your web server's root web folder, or into your javascript folder if you have a special folder for javascript code. We recommend you copy the tdAutoSelectBox folder to the root web folder so you can easily reference the javascript code using the absolute path to tdAutoSelectBox.

Once tdAutoSelectBox folder has been installed, test to make sure it works by opening the demo page. Go to http://www.yourdomain.com/tdAutoSelectBox/demo/index.html and verify that the AutoSelectBoxes are working. If you copied the files to a different folder than your root folder, adjust the URL accordingly (you will also need to modify the tdAutoSelectDefaults.js file as well -- see the next section for information). If you have a problem, please see the Troubleshooting section at the end of this document.

Integrating tdAutoSelectBox on your web pages

Once you have confirmed that tdAutoSelectBox demo page is working, you can proceed with integrating the AutoSelectBoxes in your own web pages.

To do this, add the following lines to the header area of any web pages that will make use of tdAutoSelectBox:

```
<link rel="stylesheet" type="text/css" href="/tdAutoSelectBox/css/tdStyles.css">
<script type="text/javascript" src="/tdAutoSelectBox/tdCode/tdAutoSelectBox.js"></script>
<script type="text/javascript" src="/tdAutoSelectBox/tdCode/tdAutoSelectDefaults.js"></script>
```

Then, simply change your original select boxes from this:

```
<select size="1" name="AnimalList">
<option value=""></Option>
<option value="1">Cat</Option>
<option value="2">Dog</Option>
<option value="3">Horse</Option>
<option value="4">Wolf</Option>
<option value="5">Fox</Option>
<option value="6">Lion</Option>
<option value="7">Monkey</Option>
```

To this:

```
<div id="AnimalList"></div>
<script>
var AnimalSelectBox = new Object();

AnimalSelectBox.name = "AnimalSelectBox";
AnimalSelectBox.selectBoxID = "AnimalList";
AnimalSelectBox.valList = "Cat;1|Dog;2|Horse;3|Wolf;4|Fox;5|Lion;6|Monkey;7";

createSelectBox(AnimalSelectBox)
</script>
```

And you're done! Don't forget to put the DIV tag, as shown, and make sure the div ID matches the selectBoxID attribute exactly. Also, the object.name must match as well, for instance, *AnimalSelectBox.name = "AnimalSelectBox" .*

If you encounter any problems, please consult the Troubleshooting section.

In the next section, we will show how to use tdAutoSelectBox's more advanced functions and how to customize it for your needs.

Using the AutoSelectBox

The DIV tag

Every AutoSelectBox must have a related DIV tag somewhere on the page. This tag must precede the <script> code, unless the <script> code is called by a function. You could have all your <script> code for controlling the AutoSelectBox together at the bottom of the page, as long as the DIV tags precede the execution of that code. The DIV tag must be unique for each AutoSelectBox on the page, and whatever name you choose must be reflected in the *selectBoxName* variable.

The tag should look something like this:

```
<div id="SelectBoxName"></div>
```

You may put text, images, or anything else between the opening and closing DIV tags, but please understand that the text will vanish when the AutoSelectBox is created.

The AutoSelectBox Object

The AutoSelectBox Object is simply a javascript object which contains numerous attributes, which is then passed into the **createSelectBox** function to generate the actual AutoSelectBox.

To create the object, simply chose a name (such as "MySelectBoxObj") and declare it as an object. You may name the AutoSelectBox Object anything you like.

```
var MySelectBoxObj = new Object();
```

Then add the required attributes:

```
MySelectBoxObj.name = "MySelectBoxObj"; //this must match the obj name
MySelectBoxObj.selectBoxID = "SelectBox1"; //this must match the select box ID
MySelectBoxObj.valList = "Cat;1|Dog;2|Horse;3|Wolf;4|Fox;5|Lion;6|Monkey;7";
```

Optionally, you may add one or more optional attributes if you like:

```
MySelectBoxObj.maxwidth = "100";
MySelectBoxObj.maxchar = "20";
MySelectBoxObj.maxheight = "50";
MySelectBoxObj.autotrunc = true;
MySelectBoxObj.freetype = false;
MySelectBoxObj.defaultName = "Cat"
MySelectBoxObj.defaultValue = "1"
MySelectBoxObj.onSelect = "alert('hi there!')";
```

Finally, pass the object into the createSelectBox function.

```
createSelectBox(MySelectBoxObj);
```

AutoSelectBox Attributes

Only three attributes are required: *name*, *selectBoxID*, and *valList*. These attributes must be set using the exact spelling and capitalization, as shown. Setting *MySelectBoxObj.selectBoxID* is not the same as setting *MySelectBoxObj.selectboxid* and will result in an error. In addition, there are several optional attributes which you may set as well. All optional attributes have default values associated with them, which are set in the tdAutoSelectDefaults.js file. You may change these default values to whatever you wish, but you can (and should) override each one on a per-needed basis, has shown in the earlier example.

.name

The *name* attribute must match exactly the name of the AutoSelectBox Object itself. For example, if you named your AutoSelectBox Object "MySelectBoxObj" then you must set *MySelectBoxObj.name = "MySelectBoxObj"*. Though this may seem odd, the limitations of javascript require this to be done in order for tdAutoSelectBox to function properly. Once again, name must match exactly the *name* of the AutoSelectBox Object, including capitalization, and it must be in double-quotes.

```
MySelectBoxObj.name = "MySelectBoxObj";
```

.selectBoxID

This is the ID of the select box, as far as tdAutoSelectBox and web page is concerned. This attribute must match the ID of the DIV field where the AutoSelectBox will appear. In HTML terms, this name is also the same as <SELECT NAME="nameOfSelectBox">, so in the case of a form, the *selectBoxID* is also the name of the form element that stores the value that the user selects. Submitting a form to another page or server-side script, you would find the value of the AutoSelectBox using whatever value you choose for this attribute. You must put the name in quotes and end the line with a semi-colon, like this:

```
MySelectBoxObj.selectBoxID = "AnimalList";
```

.valList

This is the list of values, created as a delimited string. Each complete element in the list must be separated by a pipe character (the | which is usually above the Enter key). Each element also has a Name and a Value, separated by a semi-colon (;), with the name first, and the value second. If the value is the same as the name, you do not need to enter both. The reason we have separated names and values is because the elements in a select box can have a value, such as a number like 1, 2, or 3, while displaying a visible name a user can understand, such as "Cat", or "Dog", or "Horse".

For example, if the names of items in your select box are the same as the values as well, create the *valList* like this:

```
MySelectBoxObj.valList = "Cat|Dog|Horse|Wolf|Fox|Lion|Leopard|Monkey";
```

If each name has a distinct value, create the *valList* like this:

```
MySelectBoxObj.valList = "Cat;1|Dog;2|Horse;3|Wolf;4|Fox;5|Lion;6|Leopard;7|Monkey;8";
```

Be sure to close the string with a quote and end the line with a semi-colon as shown.

.maxwidth
This sets the maximum width of the select box in pixels. You may have to determine the value of this number on your own by trail-and-error, but ultimately, the AutoSelectBox needs to be at least as long as the longest visible item in the drop-down list. You must put the value in quotes and end the line with a semi-colon. Do not add any other characters like "px", "pixels", or "%".

```
MySelectBoxObj.maxwidth = "80";
```

.maxchar
This sets the maximum number of characters that can be typed in to the text region of the AutoSelectBox. This cannot be less characters than the longest visible item in the drop-down list, but it can be greater. If you are not sure what you need for *maxchar*, just select an arbitrarily large value, like "100". You must put the value in quotes and end the line with a semi-colon.

```
MySelectBoxObj.maxchar = "30";
```

.maxheight
This sets the maximum height of the AutoSelectBox's drop-down box, in pixels. You can set this to a very small value, like "10" or "12" to just show one item at a time, or to "100" or more to show multiple lines. You may set this to "0" if you do not want the drop-down box to appear at all. You may have to experiment to get the hight of the drop-down box to something which suites your needs. You must put the value in quotes and end the line with a semi-colon. Do not add any other characters like "px", "pixels", or "%".

```
MySelectBoxObj.maxheight = "110";
```

.autotrunc
The *autotrunc* (short for AutoTruncate) option basically means that, if **true**, the AutoSelectBox list will dynamically trucate itself as the user types, shortening the list to the first matching value in the list. Clearing the AutoSelectBox will, of course, return the list to normal. If a*utotrunc* is **false**, it will show the complete select box list from start to end, and will not dynamically truncate the list as the user types. You may wish to experiment to see which way you like best. The value may be either **true** or **false**, and no other value is valid. Do **not** put the value in quotes.

```
MySelectBoxObj.autotrunc = true;
```

.freetype
By default, the AutoSelectBox will only allow the user to type values that match items listed in the drop-down box (those items listed on the value array). The *freetype* attribute can change this behavior. If set to **true**, the AutoSelectBox will allow the user to type anything he wishes into the AutoSelectBox. Generally, this is not advisable, as a user could enter data the form or application does not expect. However, for some forms and applications, this may be needed. Even so, the user can only type as many characters as equal the *maxchar* attribute.

```
MySelectBoxObj.freetype = true;
```

.defaultName
The *defaultName* is whatever visible name you would like the AutoSelectBox to be set to by default. Generally, you can leave this attribute alone, as it defaults to a blank space. However, you can set it to anything you like, even some arbitrary text such as "please select below".

```
MySelectBoxObj.defaultName = "Horse";
        or
MySelectBoxObj.defaultName = "(select below)";
```

.defaultValue
The *defaultValue* is whatever value (not the visible name) you would like the AutoSelectBox to be set to, by default. Generally, you can leave this attribute alone, as it defaults to a blank space. If you are using numeric values, you might set this to be "0", though you can set it to anything you like.

```
MySelectBoxObj.defaultValue = "3";
```

.onSelect
This attribute allows you to execute a certain other function (such as a javascript Submit or Alert) when an item in the AutoSelectBox is selected. You must pass it in like this "someFunction();" (in quotes). If you have to use quotation marks in the function, use the single tick, like so, "alert('hi there');". You may also string more than one javascript function together, like so, "alert('hi there');submit();". However, we recommend you create an external javascript function, such as *MyFunction*, and simply invoke that, instead of a single long string of javascript code.

```
MySelectBoxObj.onSelect = "alert('Hello!');";
        or
MySelectBoxObj.onSelect = "MySpecialFunction();";
```

If you need to reference the DOM objects of the AutoSelectBox for your javascript function, as you may well need to do when writing advanced javascript and AJAX code, please read the section "Understanding the DOM Objects" below. If you are not a programmer, do you not need to worry about this.

Full Example Code

Now that you know more about the attributes and what they mean, you should be able to create a fairly advanced AutoSelectBox using arrays, default values, and onSelect javascript.

See below for a few examples. You should paste these into a test page and play around with the attributes to get a feel for what they can do.

```
<div id="AnimalList"></div>
<script>
var AnimalSelectBox = new Object();
AnimalSelectBox.name = "AnimalSelectBox";
AnimalSelectBox.selectBoxID = "AnimalList";
AnimalSelectBox.maxwidth = "80";
AnimalSelectBox.maxchar = "20";
AnimalSelectBox.maxheight = "90";
AnimalSelectBox.autotrunc = true;
AnimalSelectBox.freetype = false;
AnimalSelectBox.valList = "Cat;1|Dog;2|Horse;3|Wolf;4|Fox;5|Lion;6|Monkey;7";
AnimalSelectBox.defaultValue = "3";
AnimalSelectBox.defaultName = "Horse";

createSelectBox(AnimalSelectBox);
</script>
```

We can also create a new function, called "popupTest" to test out the more advanced capabilities of the *onSelect* attribute. Try out this code:

```
<div id="AnimalList"></div>
<script>
function popupTest(selectBoxName) {
   selectBoxEl = document.getElementById("v"+selectBoxName);
   alert('You selected a ' + selectBoxEl.value);
   return true;
}
var AnimalSelectBox = new Object();
AnimalSelectBox.name = "AnimalSelectBox";
AnimalSelectBox.selectBoxID = "AnimalList";
AnimalSelectBox.maxwidth = "80";
AnimalSelectBox.maxchar = "20";
AnimalSelectBox.maxheight = "90";
AnimalSelectBox.autotrunc = true;
AnimalSelectBox.freetype = false;
AnimalSelectBox.valList = "Cat;1|Dog;2|Horse;3|Wolf;4|Fox;5|Lion;6|Monkey;7";
AnimalSelectBox.defaultValue = "3";
AnimalSelectBox.defaultName = "Horse";
AnimalSelectBox.onSelect = "popupTest('AnimalList')";

createSelectBox(AnimalSelectBox);
</script>
<button onclick="popupTest('AnimalList')">check</button>
```

The <button> tag is a button, similar to a submit button, which allows the user to execute the script without having to select an item in the list, i.e., if he simply types it in.

Customizing the CSS File

The default cascading style sheet (CSS) is located under the /css folder and is named tdStyles.css. In this folder you will also find two GIF images, *clear.gif* (a single pixel transparent image, used for spacing) and *selectdownarrow.gif* (the selectbox down arrow image). You may replace or customize the *selectdownarrow.gif* if you like, but we recommend you keep its size to 17x18 pixels.

The tdStyles.css file contains all the styling information for tdAutoSelectBox. You must either include this file on all your web pages that will use tdAutoSelectBox, or import the styles into your site's style sheet. Failure to do this will result in breakage of the AutoSelectBoxes. This is the #1 cause of problems people have been reporting. The #2 cause of problems which people are reporting are caused by a conflict between their site's default style sheet and the styles in the tdStyles.css. If you suspect a style conflict, create a simple test page using tdStyles.css and the sample code above, and see if the AutoSelectBoxes are working on your site. Then import your site's default style sheet and see if breakage occurs. If so, you can then work to narrow down which style is causing the problem. You may have to modify your site's stylesheet or enter additional style information into the tdStyles.css to compensate.

Before you modify the styles in the tdStyles.css, we recommend that you copy the .css and rename it (for example, MyStyles.css) – that way, if you make a mistake, you can always go back to the original. The purpose of relegating all display and style information to a CSS is to make it easier for you to configure tdAutoSelectBox to match your web site's existing styles, fonts, and colors. If you modify nothing else in the stylesheet, we recommend you at least change the various fonts and colors to match your web site's fonts and colors. Please keep in mind that altering some style attributes could break tdAutoSelectBox. For example, removing **overflow**, **display**, **position**, and **float** attributes will most certain cause problems. Changing **padding** and **margin** may also cause some alignment issues.

Comments have been placed throughout the style sheet to describe what each style controls. However, modification of the CSS code does require signification knowledge of the CSS standard. This is for advanced programmers only, or for those who are not afraid to learn on their own. If you need to find information on CSS coding, please visit this site: http://www.w3schools.com/css/

Understanding the DOM Objects

If you are an advanced javascript programmer, you should be aware of the structure of the standard DOM (Document Object Model) used for XML and HTML documents. tdAutoSelectBox uses DOM control to create and manipulate the various objects implemented for the AutoSelectBoxes.

We understand that you may need to write custom javascript code to interact with the various AutoSelectBox objects. An example of such interaction can be seen in the previous **onSelect** attribute which pops up the name of the selected animal.

```
<div id="AnimalList"></div>
<script>
function popupTest(selectBoxName) {
    selectBoxEl = document.getElementById("v"+selectBoxName);
    alert('You selected a ' + selectBoxEl.value);
    return true;
}
.........
<button onclick="popupTest('AnimalList')">check</button>
```

As you can see from this code, clicking the button (or using the *onSelect* attribute) will initiate a javascript function called "popupTest" with the attribute being the selectbox name. However, internal to tdAutoSelectBox, we rename the object which holds the actual value of the select box to begin with a "v" ("v" for value), as the line "*document.getElementById("v"+selectBoxName)*" would suggest. Ordinarily, you would not have known to prepend the "v" to retrieve or modify the value of the selectbox. In order to facilitate advanced javascripting and AJAX code, we will define what these objects are below. As always, to reference an object, you should use the standard *document.getElementById* javascript method.

selectBoxName: The DOM ID of the DIV where the AutoSelectBox will be created. If we call our selectbox "MySelectBox" then you can call *document.getElementById("MySelectBox")* to reference it. Ordinarily, you will not wish to modify this object directly, unless you wish to hide and reveal the entire selectbox using the *.style.display='block'* and *.style.display='none'* attributes.

v + selectBoxName: The DOM ID of the input box for the AutoSelectBox (i.e., the text box where the user can type). This holds the visible field that is shown on the AutoSelectBox. If you need to retrieve or change the visible value on the AutoSelectBox you would reference it like this: *document.getElementById("vMySelectBox")* or *document.getElementById("v" + selectBoxName)*

h + selectBoxName: The DOM ID of the hidden field which contains the actual value for the AutoSelectBox (which may be different from the visible field). If you need to retrieve or change the hidden value of the AutoSelectBox, you would reference it like this: *document.getElementById("hMySelectBox")* or *document.getElementById("h" + selectBoxName)* Please note that although the ID of the hidden value for the AutoSelectBox begins with an "h", the HTML *name* attribute is the same as the **selectBoxName** itself. Therefore, as far as the HTML form is concerned (and sending the data via GET or POST), the AutoSelectBox's value can still be found on the form element named **selectBoxName***.*

p + selectBoxName: The DOM ID of the drop-down (pop-up) list for the AutoSelectBox (i.e., scroallable list of items in the AutoSelectBox). If you need to retrieve or change the attributes on the drop-down list, you would reference it like this: *document.getElementById("pMySelectBox")* or *document.getElementById("p" + selectBoxName)*. Accessing this object might be useful if you have a need to dynamically adjust the width or hight of the drop-down list (*.style.width* or *.style.height*), or to dynamically hide and reveal the drop-down list (*.style.display*).

t + selectBoxName: The DOM ID of the actual table inside the drop-down (pop-up) list, the rows of which contain the values shown in the drop-down list. There is probably no reason why you would need to access this table, however, if you do, the attributes inside the drop-down list table can be reference like this: *document.getElementById("tMySelectBox")* or *document.getElementById("t" + selectBoxName)*. If you just need to rebuild the drop-down list, a far easier way would be to re-call the **createSelectBox** function (see "Using AJAX Functions" for an example of this).

multiBoxID: The DOM ID of the DIV where the MultiSelectBox will be created. If we call our multi-selectbox "MyMultiBox" then you can call d*ocument.getElementById("MyMultiBox")* to reference it. Ordinarily, you will not wish to modify this object directly, unless you wish to hide and reveal the entire MultiSelectBox using the *.style.display='block'* and *.style.display='none'* attributes.

m + multiBoxID: The DOM ID of the table inside the multi-box, the rows of which contain the values shown in the multi-box list. If you need to access the the attributes inside the multi-box table can be reference like this: *document.getElementById("tMySelectBox")* or *document.getElementById("t" + selectBoxName)*. If you just need to rebuild the multi-box list, a far easier way would be to re-call the **createMultiSelectBox** function.

multiBoxID_row_i: Items are stored in the multi-box as hidden checkboxes, and each item is indexed by *multiBoxID_row_i*. In other words, the forth item added to a multi-box named *MyMultiBox*, could be access by: *document.getElementById("MyMultiBox_row_4")*. Or we could loop, incrementing from 1 to i, calling *document.getElementById("MyMultiBox_row_"+i)*. Once we have the object, we can use the *.checked* and *.value* attributes to determine if the multi-box item has been checked (if *true*) or not (if *false*), and the value that will be passed to the form when it is submitted.

Using the Multi-SelectBox

About the Multi-SelectBox

The Multi-SelectBox feature of tdAutoSelectBox addresses the need to replace the traditional multi-line selection list with multiple option. In HTML, a traditional select box can be made into a multiple selection list like this:

```
<select size="5" multiple name="AnimalList">
<option value="1" selected>Cat</Option>
<option value="2">Dog</Option>
<option value="3">Horse</Option>
<option value="4" selected>Wolf</Option>
<option value="5">Fox</Option>
<option value="6">Lion</Option>
<option value="7">Leopard</Option>
<option value="8">Monkey</Option>
```

This allows the end-user to see multiple options at a time (in this case, five) and he can select more than one by holding down the mouse and selecting a sequence of options, or Ctrl-clicking individual options one by one to select non-sequential options. This has three disadvantages: 1) finding items in long lists can be cumbersome, 2) Ctrl-clicking to select and de-select is far from being intuitive, and 3) one mis-click could unselect everything which had been selected before. If a user spends five minutes hunting and Ctrl-clicking for items in a long list, then mistakenly clicks without Ctrl-clicking, all previously select items become unselected!

The Multi-SelectBox feature of tdAutoSelectBox remedies these issues, while at the same time leveraging the power of the AutoSelectBox. Placing an AutoSelectBox with the Multi-SelectBox option on a web page will present the user with an AutoSelectBox (as described in the previous sections), below which will be a small, expandable box to hold all selected values. To the right of the SelectBox the user will see two buttons, one to "add" a selected item, the other to "del" (delete) an item. Using the Multi-SelectBox is quite simple: the user types the item name or selects it from the list, as usual, and then he may click "add". This will copy the item to the box below, which holds all selected values.

The Multi-Select DIV tag

When using a Multi-Select Box, you will have to have a DIV tag for both the AutoSelectBox (as usual) as well as a second DIV for the multi-select box. Generally, you would place these one after the other, but you are not required to do so. However, the tags must precede the <script> code, unless the <script> code is called by a function. The DIV tag must be unique for each Multi-SelectBox on the page, and whatever name you choose must be reflected in the **multiBoxID** variable.

Together, the two tag should look something like this:

```
<div id="ClothesSelectBox"></div>
<div id="ClothesMultiList"></div>
```

The Multi-SelectBox Object

The Multi-SelectBox Object is simply a javascript object which contains numerous attributes, which is then passed into the **createMultiSelectBox** function to generate the Multi-SelectBox.

To create the object, simply chose a name (such as "MyMultiSelectBoxObj") and declare it as an object. You may name the MultiSelectBox Object anything you like.

```
var MyMultiSelectBoxObj = new Object();
```

Then add the required attributes:

```
MyMultiSelectBoxObj.name = "MyMultiSelectBoxObj"; //this must match the obj name
MyMultiSelectBoxObj.multiBoxID = "MultiBox1"; //this must match the multi select box ID
MyMultiSelectBoxObj.multiwidth =  MySelectBoxObj.maxwidth;
MyMultiSelectBoxObj.multiDefList = "Cat;1|Wolf;4";
```

Finally, pass the object into the createMultiSelectBox function, along with the AutoSelectBox Object as well.

```
createMultiSelectBox(MyMultiSelectBoxObj, MySelectBoxObj);
```

Multi-SelectBox Attributes

Only two attributes are required: *name* and *multiBoxID*. These attributes must be set using the exact spelling and capitalization, as shown. Setting *MyMultiSelectBoxObj.multiBoxID* is not the same as setting *MyMultiSelectBoxObj.multiboxid* and will result in an error. In addition, there are several optional attributes which you may set as well. All optional attributes have default values associated with them, which are set in the tdAutoSelectDefaults.js file. You may change these default values to whatever you wish, but you can (and should) override each one on a per needed basis, has shown in the earlier example.

.multiBoxID

This is the ID of the Multi-SelectBox, as far as tdAutoSelectBox and the web page is concerned. The name must match the ID of the DIV field where the Multi-SelectBox will appear. In HTML terms, this name is the same as <SELECT NAME="nameOfMultiSelectBox">, so in the case of a form, the multiBoxID is the name of the table that stores all the values that the user has selected. Submitting a form to another page or server-side script, you would find the values of the MultiSelectBox using whatever name you choose for this attribute. You must put the name in quotes and end the line with a semi-colon, like this:

```
MyMultiSelectBoxObj.multiBoxID = "ClothesMultiList";
```

.multiwidth

Generally, the width of the Multi-SelectBox should be the same as the width of the AutoSelectBox defined earlier (the **maxwidth** attribute). Therefore, you can simply set **multiwidth** equal to **maxwidth**. However, you may set this width to anything you like. If so, then you must put the value in quotes and end the line with a semi-colon. Do not add any other characters like "px", "pixels", or "%".

```
MyMultiSelectBoxObj.multiwidth = MySelectBoxObj.maxwidth;
```

.multiDefList
The **multiDefList** holds one ore more values that will be shown inside the multi-selectbox by default. Most Multi-SelectBoxes will not have a default value (though the associated AutoSelectBox might). However, this gives you the option to add pre-set values if you so desire.

The **multiDefList** should be created as a delimited value list, just like the delimited values in the **valList**. Each element in the list must be separated by a pipe character (the | which is usually above the Enter key). Each element has a Name and Value, separated by a simi-colon (;). If the value is the same as the name, you do not need to enter both.

For example, if the names of items in your select box are the same as the values as well, create the **multiDefList** like this:

```
MyMultiSelectBoxObj.multiDefList = "Shirt|Boots|Jacket";
```

If each name as a distinct value, create the **multiDefList** like this:

```
MyMultiSelectBoxObj.multiDefList = "Shirt;1|Boots;4|Jacket;9";
```

Be sure to close the string with a quote and end the line with a semi-colon as shown.

Full Example Code

Now that you know more about the attributes and what they mean, you should be able to create a Multi-SelectBox paired with an AutoSelectBox. See the following examples. You should paste this into a test page and play around with the attributes yourself.

```
<div id="ClothesSelectBox"></div>
<div id="ClothesMultiList"></div>
<script>
//shows how to use the multi-select box features

var ClothesSelectBox = new Object();
ClothesSelectBox.name = "ClothesSelectBox";
ClothesSelectBox.selectBoxID = "ClothesSelectBox";
ClothesSelectBox.maxwidth = "100";
ClothesSelectBox.maxchar = "20";
ClothesSelectBox.autotrunc = false;
ClothesSelectBox.freetype = false;
ClothesSelectBox.valList = "Shirt|Shoe|Pants|Boots|Belt|Dress|Hat|Gloves|Jacket|Night Gown";

var ClothesMultiSelectBox = new Object();
ClothesMultiSelectBox.name = "ClothesMultiSelectBox";
ClothesMultiSelectBox.multiBoxID = "ClothesMultiList";
ClothesMultiSelectBox.multiwidth = ClothesSelectBox.maxwidth;
ClothesMultiSelectBox.multiDefList = "Shirt|Boots|Jacket";

createSelectBox(ClothesSelectBox);
createMultiSelectBox(ClothesMultiSelectBox, ClothesSelectBox);
</script>
```

Using the AJAX Functions

AJAX, which is an abbreviation for Asynchronous Javascript And XML, is a collection of technologies that enables developers to design HTML web page that can send and receive information to and from a server-side script (such as a database querying engine), without having to refresh the page. This is enormously advantageous, as it can vastly improve the user's experience when dealing with interactive pages.

tdAutoSelectBox now supports these advanced AJAX capabilities. With just a few lines of code, it is possible for a tdAutoSelectBox to asynchronously send and receive information from a server-side script (.aspx, .php, .jsp, .pl, etc.), which may process the request, query a database for data, process the data, and return the data to the page.

> This feature is only for advanced programmers, as you will have to implement the server-side script and a javascript response handler yourself. However, if you follow the example below you may get some idea of how this is done.

Built-in AJAX Functions

To facilitate and simplify AJAX calls, tdAutoSelectBox comes with several additional functions. You may use your own AJAX functions if you so desire. These are here simply for your convenience.

initRequestor()
This function returns a new XMLHTTPRequest object. It will decide which type of XMLHTTPRequest object to send back, depending on the client's browser. XMLHTTPRequest is supported on all modern browsers (I.E. 5.5+, Mozilla Firefox, Netscape 8, Opera 8).

waitForReadyState(req, responseFunction)
This function is called to monitor the "ready state" of the server-side script. It takes two parameters: *req*, which is the XMLHTTPRequest object created above, and *responseFunction*, which is the function **waitForReadyState** should invoke upon completion. When invoked, it will wait until status message 200 is returned from the server-side script, which means the server-side script has completed successfully. Otherwise, if some other state is returned (such as 404 page not found, or 500 server error), that message will be displayed in a javascript alert box. When **waitForReadyState** receives message 200, it will invoke the *responseFunction* that was passed in.

parameterizeFormElements(formObj)
This function will convert all form elements into parameters to pass into the server-side script. This is a convenience function – you may not need to use it. In the event that you need to submit an entire form and all elements within that form to a server-side script, you may need to convert the values of that form element to URL parameters. **parameterizeFormElements** takes the *formObject* as a parameter (i.e., document.forms.MyFormName), and returns a URL encoded string of parameters. These parameter can then be submitted with the XMLHTTPRequest object.

Process Request

The function **processRequest** is not a built-in function, and needs to be written by the web page developer to facilitate processing of the AJAX request. It may be unique to his or her page, depending on how the AJAX request is to be handled. Our example is fairly generic and can be used as a baseline for similar functions. The function does not need to be named **processRequest**, but you do need to pass in the URL of the server-side script to be invoked.

Note that the server-side script generally needs to be on your own server – XMLHTTP requests to remote servers may generate an error.

```
function processRequest(thisForm, processingPage) {

    var params = parameterizeFormElements(thisForm);

    var req = initRequestor();
    if (thisForm.id == "stateForm") {
       processResponse = processCityCodes;
    } else if (thisForm.id == "cityForm") {
       processResponse = processZipCodes;
    }

    req.onreadystatechange = waitForReadyState(req, processResponse);

    req.open("POST", processingPage, true);
    req.setRequestHeader("If-Modified-Since", "Sat, 1 Jan 2000 00:00:00 GMT");
    req.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    req.setRequestHeader("Connection", "close");
    req.send(params);

    return false; //return false to avoid automatic form submittal
 }
```

As you can see, in this example we convert the form into a list of parameters and set those parameters on the Request object. Also, we have not one, but two Process Responses to choose from (**processCityCodes** and **processZipCodes**), depending on which form was submitted (**stateForm** or **cityForm**). You may have only one Process Response to deal with, or many more. How you handle your request and response is entirely dependent on what you are trying to accomplish. The RequestHeader information is set to ensure the page is not cached by the client's browser. Page cashing is the number one problem that plagues all AJAX applications.

Process Response

In our example, we have not one, but two process response functions. These are **processCityCodes** and **processZipCodes**.

```
var CitySelectBox; //this must be declared as a global
function processCityCodes(resXML, resText) {

   CitySelectBox = new Object();
   CitySelectBox.name = "CitySelectBox";
   CitySelectBox.valList = resText;
   CitySelectBox.selectBoxID = "cityCode"
   CitySelectBox.maxwidth = "120";
   CitySelectBox.maxchar = "20";
   CitySelectBox.maxheight = "90";
   CitySelectBox.autotrunc = true;
   CitySelectBox.freetype = false;
   CitySelectBox.onSelect = "processRequest(document.forms.cityForm, 'AJAXprocessor.php');";

   //must re-create the select box
   createSelectBox(CitySelectBox);
}
```

The function **processCityCodes** is called if the **stateForm** was submitted (see above) and when **waitForReadyState** receives a 200 message from the server-side script. The purpose of this function is to convert the response (sent back from the server-side script) to a list of city codes, to be displayed in a newly created AutoSelectBox. This new AutoSelectBox has an *onSelect* parameter to again call processRequest when a city is selected, which is sent to the server-side script to find the zip codes in that city. The **processRequest** function then invokes **processZipCodes** to create the zip code Multi-SelectBox. The difference between **resXML** and **resText** is that **resXML** contains an XML formatted response, while **resText** holds a flat text string. We are using **resText** for simplicity, however, if you wish to work with advanced XML messages you may need to use **resXML**.

```
var ZipSelectBox; //this must be declared as a global
var MultiZipSelectBox; //this must be declared as a global
function processZipCodes(resXML, resText) {

   ZipSelectBox = new Object();
   ZipSelectBox.name = "ZipSelectBox";
   ZipSelectBox.valList = resText;
   ZipSelectBox.selectBoxID = "zipCode"
   ZipSelectBox.maxwidth = "60";
   ZipSelectBox.maxchar = "5";
   ZipSelectBox.maxheight = "115";
   ZipSelectBox.autotrunc = true;
   ZipSelectBox.freetype = false;

   MultiZipSelectBox = new Object();
   MultiZipSelectBox.name = "MultiZipSelectBox";
   MultiZipSelectBox.multiBoxID = "multiZip";
   MultiZipSelectBox.multiwidth = ZipSelectBox.maxwidth;

   //must re-create the select box
   createSelectBox(ZipSelectBox);

   //if multiselectbox already exists, do not re-create it
   if (document.getElementById(MultiZipSelectBox.multiBoxID).innerHTML=="") {
      //if we do want to wipe-out the old select box, remove this code
      createMultiSelectBox(MultiZipSelectBox, ZipSelectBox);
    }
}
```

This is the second process request function, **processZipCodes**. It is called when the cityCode AutoSelectBox is triggered and when **waitForReadyState** receives a 200 message from the server-side script. The purpose of this function is to convert the response from the server-side script to a list of zip codes, to be displayed in a newly created AutoSelectBox. In addition to this, a Multi-SelectBox is also created, so that more than one zip code can be added. Because we only create the Multi-SelectBox once, we can add zip codes from more than one state, together in a single Multi-SelectBox.

For a complete example of these functions in action, please view the AJAX sample code which came with your copy of tdAutoSelectBox, or run the AJAX demo at http://www.tetraktysdevelopment.com.

AutoSelectBox with AJAX

Creating the AutoSelectBox to interact with AJAX functions is only slightly different than in previous examples. The only major change is the addition of the call to **processRequest** using the *onSelect* attribute. This is how the actual **processRequest** is triggered. In our example, when the user selects a state (or clicks the State Submit button), **processRequest** is invoked. This will then call **processCityCodes** (as above), creating a new select box listing the cities in that state. As parameters on the **processRequest** call we should send the related Form and specify the server-side script that processes the AJAX request (in this case, AJAXprocessor.php).

```
<form id="stateForm">
<b>Select or type in a State.</b><br />
<div id="stateCode"></div>
<script>
var StateSelectBox = new Object();
StateSelectBox.name = "StateSelectBox";

StateSelectBox.valList = "Arizona;AZ|California;CA|Colorado;CO";

StateSelectBox.selectBoxID = "stateCode";
StateSelectBox.maxwidth = "90";
StateSelectBox.maxchar = "13";
StateSelectBox.maxheight = "110"; //pixels
StateSelectBox.autotrunc = false;
StateSelectBox.freetype = false;
StateSelectBox.onSelect = "processRequest(document.forms.stateForm, 'AJAXprocessor.asp')";

createSelectBox(StateSelectBox);
</script>
<button type="button"
 onclick="processRequest(document.forms.stateForm, 'AJAXprocessor.php');">Submit</button>
</form>
</p>
<p>
<form id="cityForm">
<b>Choose the City</b>
<div id="cityCode">(please select the state first)</div>
<button type="button"
 onclick="processRequest(document.forms.cityForm, 'AJAXprocessor.php');">Submit</button>
</form>
</p>
<p>
<form id="zipForm">
<b>Add Zip Codes:</b><br />
<div id="zipCode">(please select a city first)</div>
<div id="multiZip"></div>
<button type="button" onclick="alert('This would submit the page.');">Submit</button>
</form>
```

Looking at the code above, we can see that the AutoSelectBoxes are created the same way in previous examples. We have a DIV tag for each of the AutoSelectBoxes (**stateCode**, **cityCode**, and **zipCode**), and each one has a different Form. The only major difference is that **cityCode** and **zipCode** are created in separate functions – **processCityCodes()** and **processZipCodes()** respectively.

Server-Side Script

However, the client-side javascript code is only half the story -- the other half is the server-side code. A server-side script is any kind of program that runs on a web server, that has the ability to process requests and return a response over the Internet. Examples of server-side script include ASP, PHP, and JSP pages, as well as CGI and PERL scripts. You could, theoretically, also call an XML file or RSS file directly, and interpret the XML data on the process response javascript code.

```
<?php

//create the header
header( 'Expires: Mon, 26 Jul 1997 05:00:00 GMT' );
header( 'Last-Modified: ' . gmdate( 'D, d M Y H:i:s' ) . ' GMT' );
header( 'Cache-Control: no-store, no-cache, must-revalidate' );
header( 'Cache-Control: post-check==0, pre-check==0', false );
header( 'Pragma: no-cache' );

$AJAXResponse = "";

if(isset($_REQUEST['stateCode'])) {
   $StateCd = $_REQUEST['stateCode'];
} else {
   $StateCd = "";
}

if(isset($_REQUEST['cityCode'])) {
   $CityCd = $_REQUEST['cityCode'];
} else {
   $CityCd = "";
}

if ($StateCd != "") {
   if ($StateCd == "AZ") {
      $AJAXResponse = "Flagstaff;FLG|Phoenix;PHX|Scottsdale;SCF|Tempe;TMP|Tucson;TUS";
   } else if ($StateCd == "CA") {
       $AJAXResponse = "Fresno;FCH|Los Angeles;LAX|Oakland;OAK|Pasadena;PAS|Sacramento;SAC|San
Francisco;SFO";
   } else if ($StateCd == "CO") {
             $AJAXResponse = "Boulder;BOL|Colorado Springs;COS|Denver;DEN|Ft. Collins;FNL|Grand
Junction;GJT|Trinidad;TAD";
   } else {
      $AJAXResponse = "nothing;0";
   }
} else {
   if ($CityCd == "FLG") {
      $AJAXResponse = "86001|86002|86003|86004|86011";
   } else if ($CityCd == "PHX") {
      $AJAXResponse = "85015|85016|85038|85078|85079|85080|85082|85085|85098";
.
. //shortened for brevity
.
   } else {
      $AJAXResponse = "nothing;0";
   }
}
echo $AJAXResponse;
?>
```

The above code is an example of a server-side PHP script that can process the AJAX request and return a response formatted for tdAutoSelectBox. This code is shorten to fit on one page, but the full example PHP script can be found in the example code which comes with your copy of tdAutoSelectBox, as well as an ASP script for Microsoft web servers. To use a server-side script, your web server or hosting provider must support them. Most hosting providers now support PHP, PERL, and CGI. A few might support ASP or JSP. If you need to learn more about any of these scripting languages, we encourage you to search Google for programming resources and guides.

As you can see, our example uses hard-coded data. In a real application, we might make a database query to collect information based on the value selected in the select box. In either case, we should create a value list in the format which tdAutoSelectBox understands, and that string is then sent back to the requesting page to be processed in javascript. As discussed earlier, tdAutoSelectBox uses a | (pipe) to separate the individual terms in the list, and a semi-colon to separate the name/value of each term. tdAutoSelectBox will parse this into an array. However, we could also send back XML for more advanced XML processing.

Again, we must emphasize that using AJAX will require advanced programming on your part. However, we believe this functionality will bring an entirely new dimension of interactive to your tdAutoSelectBox powered applications.

Troubleshooting

I can't get tdAutoSelectBox to work!

There could be many different things going on which may contribute to tdAutoSelectBox not working on your site. The very first thing you should do is verify you have the correct path to the javascript and CSS file referenced on your web pages.

The following lines should be in the header area of any web pages that will make use of tdAutoSelectBox:

```
<link rel="stylesheet" type="text/css" href="/tdAutoSelectBox/css/tdStyles.css">
<script type="text/javascript" src="/tdAutoSelectBox/tdCode/tdAutoSelectBox.js"></script>
<script type="text/javascript" src="/tdAutoSelectBox/tdCode/tdAutoSelectDefaults.js"></script>
```

Remember to reference both tdAutoSelectBox.js and tdAutoSelectDefaults.js. The tdAutoSelectDefaults.js contains default values for various options. Though you may change these values, if tdAutoSelectDefaults.js is not included, tdAutoSelectBox will not function. If, for whatever reason, your path to the tdAutoSelectBox directory is different, or you have renamed the tdAutoSelectBox directory, please adjust the paths or folder names accordingly.

Antoher common problem is caused by not entering the attributes on the AutoSelectBox Object (or Multi-SelectBox Object) correctly. The attribute names must be entered exactly as shown in the examples, and they are case-specific (capitalization matters). Setting *MySelectBoxObj.selectBoxID* is not the same as setting *MySelectBoxObj.selectboxid* and will result in an error. Furthermore, the *name* attribute on the object must match exactly the name of the AutoSelectBox Object itself. For example, if you named your AutoSelectBox Object "MySelectBoxObj" then you must set *MySelectBoxObj.name = "MySelectBoxObj"*. Also max sure you use double-quotes correctly, and that you end each line with a semi-colon.

If you are still having problems, continue reading through this Troubleshooting section. Your answer may be addressed below.

The AutoSelectBox is working, but has display issues

Display issues are usually caused by a problem in the style sheet. Either you did not import the tdStyles.css, or you have a style sheet that changes a style uses by tdStyles.css, or visa-versa. If you suspect a style conflict, create a simple test page using tdStyles.css and the sample code above, and see if the AutoSelectBoxes are working on your site. Then import your site's default style sheet and see if breakage occurs. If so, you can then work to narrow down which style is causing the problem. You may have to modify your site's stylesheet, but more often than not, you can simply enter additional style information into the tdStyles.css to compensate.

Before you modify the styles in the tdStyles.css, we recommend that you copy the .css and rename it (for example, MyStyles.css) – that way, if you make a mistake, you can always go back to the original. The purpose of relegating all display and style information to a CSS is to make it easier for you to configure tdAutoSelectBox to match your web site's existing styles, fonts, and colors. If you modify nothing else in the stylesheet, we recommend you at least change the various fonts and colors to match your web site's fonts and colors. Please keep in mind that altering some style attributes could break tdAutoSelectBox. For example, removing **overflow**, **display**, **position**, and **float** attributes will most certain cause problems. Changing **padding** and **margin** may also cause some alignment issues.

Comments have been placed throughout the style sheet to describe what each style controls. However, modification of the CSS code does require signification knowledge of the CSS standard. This is for advanced programmers only, or for those who are not afraid to learn on their own. If you need to find information on CSS coding, please visit this site: http://www.w3schools.com/css/

I do not see the drop-down arrow for the select box

This is almost always caused when either the clear.gif and/or the selectdownarrow.gif are not referenced correctly. These two GIF files are under the /tdAutoSelectBox/css/ folder. If you have placed these GIF files in a different folder, open the tdAutoSelectDefaults.js file which can be found under /tdAutoSelectBox/xxABC/ (see previous section for what we mean by xxABC). In this file you will find a variable called **PathToImages**. Make sure **PathToImages** points to the *exact* absolute path where the clear.gif and the selectdownarrow.gif are located.

If you have in some way modified or changed either the clear.gif or the selectdownarrow.gif and they are not showing correctly, please revert back to the original image. You should be able to replace selectdownarrow.gif with any image you like, as long as it is 17x18 pixels, and uses the same name.

I am getting a javascript error

A javascript error may occur if you are not referencing the actual tdAutoSelectBox Javascript code correctly. Please make sure you are importing the the Javascript files as shown on the previous page.

If you are certain that the AutoSelectBox code is being executed correctly, but you are still getting a javascript error, make sure you follow the example given on page 4 of this document exactly as shown. If a value needs to be in quotes, put it in quotes. Verify you are setting all the required attributes on the AutoSelectBox Object correctly. If you are using *onSelect*, temporarily remove the *onSelect* attribute to see if it is the cause of the problem.

Having verified all of this, if you are still getting a javascript error on the AutoSelectBox, please email support@tetraktysdevelopment.com and send us a sample of your web page (or direct us to

the web page on the Internet), along with the name of your web browser and its version number.

The Multi-SelectBox buttons are gone, or are mis-aligned

This is possibly due to a stylesheet problem. See the previous section on Display Issues.

However, the Multi-SelectBox buttons may also be mis-aligned if you set the DIV tags for the AutoSelectBox and Multi-Select Box apart from each other. For instance, if you create the DIV tag for the AutoSelectBox in a table cell, and then place the DIV tag for the Multi-Select Box in a different table cell, the buttons could vanish, appear in a strange place, or over-lap some other element on the page. Please keep the AutoSelectBox and Multi-Select Box DIV tags together (the AutoSelectBox tag just above the Multi-Select Box tag). If you simply must have the tags side-by-side, or in some other configuration, you will need to modify the positioning attributes of the *multiBoxButton* style in tdStyles.css to accommodate this.

I see a horizontal scroll bar on the drop-down list.

This happens if you do not make the AutoSelectBox width wide enough. There is a value somewhere in the drop-down list that is longer than the AutoSelectBox width (which is set in pixels, not characters). Try to increase the size of the **maxwidth** attribute.

The *onSelect* attribute is giving javascript errors

Most likely you are using double-quotes inside the onSelect javascript, or have made some other javascript mistake.

Wrong
```
onSelect = "alert("Hello!")";
```

Right
```
onSelect = "alert('Hello!');";
```

It is also possible you may have an error in your own javascript, so be sure to double check that. If you have complex javascript, please do not write it all into the *onSelect* attribute. Rather, write an external javascript function and call it using the *onSelect* attribute, like so:

```
<div id="AnimalList"></div>
<script>
function popupTest(selectBoxName) {
     selectBoxEl = document.getElementById("v"+selectBoxName);
     alert('You selected a ' + selectBoxEl.value);
     return true;
}
.
.
.
AnimalSelectBox.onSelect = "popupTest('AnimalList')";

createSelectBox(AnimalSelectBox);
</script>
```

Some of my visitors are reporting problems with AutoSelectBox

If AutoSelectBox works fine for you when you test your website, but a handful of your users are having issues, inquire as to what browser and what version of that browser they are using. If

they are using an old or incompatible browser, point them to a newer browser, like Firefox, Opera, or IE 6.0. Also, they may have javascript turned off for some reason. AutoSelectBox can only work if the user has javascript enabled.

I think I found a bug

Believe it or not, it is always possible that a problem you have encountered is the result of a bug on our part. If you have eliminated all other troubleshooting solutions as a resolution to your issue, please email support@tetraktysdevelopment.com and send us a sample of your web page (or direct us to the web page on the Internet), along with the name of your web browser and its version number.