SCRIPT.ACULO.US - AUTO COMPLETION

Out of the box, script.aculo.us supports two sources for auto-completion −

- Remote sources *obtainedthroughAjax*,
- Local sources *stringarraysinyourwebpage′sscripts*.

Depending on the source you're planning to use, you'll instantiate *Ajax.Autocompleter* or *Autocompleter.Local*, respectively. Although equipped with specific options, these two objects share a large feature set and provide a uniform user experience.

There are four things you'll always pass to these objects while building them −

- The text field you want to make autocompletable. As usual, you can pass the field itself or the value of its id = attribute.
- The container for autocompletion choices, which will end up holding a <ul></li> list of options to pick from. Again, pass the element directly or its **id =**. This element is most often a simple <div>.</p></li>
- The data source, which will be expressed, depending on the source type, as a JavaScript array of strings or as a URL to the remote source.
- Finally, the options. As always, they're provided as a hash of sorts, and both autocompletion objects can make do with no custom option; there are suitable defaults for everything.

To use script.aculo.us's autocompletion capabilities, you'll need to load the controls.js and effects.js modules along with the prototype.js module. So, your minimum loading for script.aculo.us will look like this −

```
<script type="text/javascript" src="/javascript/prototype.js"></script>
<script type="text/javascript"  src="/javascript/scriptaculous.js?
load=effects,controls"></script>
```

Creating an Ajax Auto-Completer

The construction syntax is as follows −

```
new Ajax.Autocompleter(element, container, url [ , options ] )
```

The constructor for the *Ajax.Autocompleter* accepts four parameters −

- The element name or reference to a text field that is to be populated with a data choice.
- The element name or reference to a <div> element to be used as a menu of choices by the control.
- The URL of the server-side resource that will supply the choices.
- The usual options hash.

Options

You can use one or more of the following options while creating your Ajax.Autocompleter object.

Option	Description
paramName	The name of the query parameter containing the content of the text field that is posted to the server-side resource. Defaults to the name of the text field.

minChars	Number of characters that must be entered before a server-side request for choices can be fired off. Defaults to 1.
Frequency	The interval, in seconds, between internal checks to see if a request to the server-side resource should be posted. Defaults to 0.4.
Indicator	The id or reference to an element to be displayed while a server-side request for choices is underway. If omitted, no element is revealed.
Parameters	A text string containing extra query parameters to be passed to the server-side resource.
updateElement	A callback function to be triggered when the user selects one of the choices returned from the server that replaces the internal function that updates the text field with the chosen value.
afterUpdateElement	A callback function to be triggered after the updateElement function has been executed.
Tokens	A single text string, or array of text strings that indicate tokens to be used as delimiters to allow multiple elements to be entered into the text field, each of which can be auto-completed individually.

Example

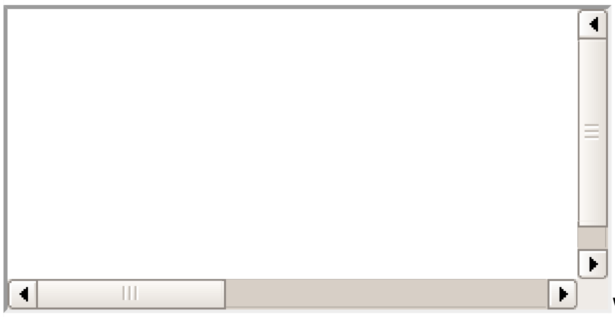
```
<html>
   <head>
      <title>Simple Ajax Auto-completer Example</title>

      <script type="text/javascript" src="/javascript/prototype.js"></script>
      <script type="text/javascript"  src="/javascript/scriptaculous.js?
load=effects,controls"></script>
      <script type="text/javascript">

         window.onload = function() {

            new Ajax.Autocompleter( 'autoCompleteTextField', 'autoCompleteMenu',
'/script.aculo.us/serverSideScript.php', {});
         }
      </script>
   </head>
   <body>

      <p>Type something in this box and then select suggested option from the list </p>

      <div>
         <label>Text field:</label>
         <input type="text" />
         <div ></div>
      </div>

   </body>
</html>
```

Now, we need a server side to access this page and serve the data source URL *serverSideScript. php*. You will keep a complete logic to display suggestions in this script.

Just for example, we are keeping a simple HTML text in *serverSideScript.php*. You can write your script using CGI, PHP, Ruby, or any other server side scripting to choose appropriate suggestions and format them in the form of <ul><li>...</li></ul> and pass them back to the caller program.

```
<ul>
   <li>One</li>
   <li>Two</li>
   <li>Three</li>
   <li>Four</li>
   <li>Five</li>
   <li>Six</li>
</ul>
```

This will produce following result −


with different options discussed in the above table.

Creating a Local Auto-Completer

Creating a local auto-completer is almost identical to creating an Ajax Auto-completer as we have discussed in the previous section.

The major difference lies in how the backing data set to use for auto-completion is identified to the control.

With an Ajax Auto-completer, we have supplied the URL of a server-side resource that would perform the necessary filtering, given the user input, and return only the data elements that matched. With a Local Autocompleter, we supply the full list of data element instead, as a JavaScript String array, and the control itself performs the filtering operation within its own client code.

The whole construction syntax is actually as follows −

```
new Autocompleter.Local(field, container, dataSource [ , options ] );
```

The constructor for the Autocompleter.Local accepts four parameters −

- The element name or reference to a text field that is to be populated with a data choice.
- The element name or reference to a <div> element to be used as a menu of choices by the control
- For the third parameter, instead of a URL as with the server-assisted auto-completer, we supply a small String array, which contains all of the possible values.
- The usual options hash.

Options

You can use one or more of the following options while creating your Autocompleter.Local object.

Option	Description
Choices	The number of choices to display. Defaults to 10.
partialSearch	Enables matching at the beginning of words embedded within the completion strings. Defaults to true.
fullSearch	Enables matching anywhere within the completion strings. Defaults to false.
partialChars	Defines the number of characters that must be typed before any partial matching is attempted. Defaults to 2.
ignoreCase	Ignores case when matching. Defaults to true.

Example

```
<html>
   <head>
      <title>Simple Ajax Auto-completer Example</title>
		
      <script type="text/javascript" src="/javascript/prototype.js"></script>
      <script type="text/javascript"  src="/javascript/scriptaculous.js?
load=effects,controls"></script>
      <script type="text/javascript">
		
         window.onload = function() {
			
            new Autocompleter.Local('autoCompleteTextField', 'autoCompleteMenu',
['abcdef','abcdeg','xyzabcefg', 'tybabdefg','acdefg'], {ignoreCase:false});
         }
      </script>
   </head>
   <body>
	
      <p>Type something in this box and then select suggested option from the list </p>

      <div>
         <label>Text field:</label>
         <input type="text" />
         <div ></div>
      </div>
		
   </body>
</html>
```

When displayed, and after the character 'a' is typed into the text box, it displays all the matching options.

Use our online compiler for a better understanding of the code with different options discussed in the above table.

This will produce following result −

Loading [MathJax]/jax/output/HTML-CSS/jax.js